INAUGURAL-DISSERTATION

zur
Erlangung der Doktorwürde
der
GESAMTFAKULTÄT FÜR MATHEMATIK, INGENIEUR- UND NATURWISSENSCHAFTEN
der
RUPRECHT-KARLS-UNIVERSITÄT
HEIDELBERG

vorgelegt von

Subhash Chandra Pujari (M.Sc.)

aus Nainital

Tag der mündlichen Prüfung:

Neural Patent Classification beyond Title and Abstract: Leveraging Patent Text and Metadata

Supervisor: Prof. Dr. Michael Gertz

Abstract

Intellectual property violations involve substantial litigation and license costs, because of which patent search is of utmost importance. Over the years, patent corpora have amassed millions of patents, making manual searches impractical. Patent classification techniques help domain experts to search and analyze patents. On submission to an examination office, a patent application is assigned with labels from pre-defined patent taxonomies, e.g., Cooperative Patent Classification (CPC) and International Patent Classification (IPC). CPC/IPC classification helps to route patent applications to the correct department and assists in performing prior art searches. In addition to CPC/IPC classification, we address the classification task associated with the Patent Landscape Study (PLS), a process that allows organizations to search patents, categorize them by custom labels, and analyze them to derive crucial insights. This thesis significantly contributes to the improvement of patent classification systems by addressing the key challenges described below.

Most of the existing CPC classification datasets provide only limited texts of the included patents and are, therefore, insufficient for our experiments. In response to this issue, we release a CPC classification dataset that includes the full texts of patents. Further, the unavailability of open-source datasets is a major bottleneck for the automation of PLS. To address this challenge, we curate, enrich, and release three open-source datasets from two diverse domains.

Despite CPC/IPC classification being a hierarchical multi-label classification task, most prior neural network models have not considered the hierarchical taxonomy when designing model architectures and have often predicted labels only for a single level. We make a major contribution with our memory-efficient model architecture, which shares a single transformer-based language model across multiple classification heads, one for each label in the taxonomy, and uses hierarchical links in the model architecture. We demonstrate that the proposed technique consistently outperforms baselines, particularly for infrequent labels.

Our analysis shows that the sentences and abstracts of patents are often duplicated, illustrating the relevance of the full texts of patents to perform classification. However, transformer-based language models that take 512 or 4,096 tokens as input are insufficient for patents, which contain 12.5k tokens on average. Motivated by these factors, we make a major contribution with our document representation technique, which combines truncated section text embeddings using vector summation, performing better than baselines. In addition, we propose a sentence ranker and demonstrate that the extractive summarization techniques effectively select informative sentences for patent classification.

Unlike CPC/IPC classification, in the case of PLS, the CPC/IPC labels are known during inference. As a major contribution, we enrich the document representation by combining CPC/IPC labels with patent text to predict PLS-oriented categories, often representing concepts different from CPC/IPC labels. To demonstrate the broader applicability of the proposed technique, we apply it to a similar task: classifying research publications into target categories using text and author-provided keywords as input.

Zusammenfassung

Mit Verletzungen des geistigen Eigentums sind hohe Prozess- und Lizenzkosten verbunden, was die Patentsuche so eminent wichtig macht. Im Laufe der Jahre haben sich Millionen Patente in Patentkorpora angesammelt, was eine manuelle Suche erschwert. Verfahren zur Patentklassifikation unterstützen Sachkundige bei der Suche und Analyse von Patenten. Nach Einreichung beim Patentamt wird einem Patentantrag Klassen aus vordefinierten Patenttaxonomien zugeordnet, zum Beispiel die gemeinsame Patentklassifikation (CPC) oder die internationale Patentklassifikation (IPC). Diese CPC-/IPC-Klassifikation hilft dabei, den Patentantrag an die richtige Abteilung weiterzuleiten und eine Recherche zum Stand der Technik durchzuführen. Neben der CPC-/IPC-Klassifikation beschäftigen wir uns mit der Klassifikationsaufgabe, die mit der Patentanalyse (Patent Landscape Study, PLS) verknüpft ist, einem Prozess, der es Unternehmen ermöglicht, relevante Patente zu suchen, zu kategorisieren und zu analysieren. Diese Dissertation leistet einen maßgeblichen Beitrag zu besseren Patentklassifikationssystemen, indem wir die im Folgenden beschriebenen zentralen Herausforderungen angehen.

Ein großes Hindernis für die Automatisierung der PLS ist die fehlende Verfügbarkeit von öffentlich zugänglichen Datensätzen. Dieses Problem adressieren wir, indem wir drei öffentlich zugängliche Datensätze aus zwei unterschiedlichen Domänen kuratieren, erweitern und veröffentlichen. Darüber hinaus liefern die meisten vorhandenen Datensätze zur CPC-/IPC-Klassifikation nur einen Ausschnitt der Patente, sie kommen daher für unsere Experimente nicht infrage. Mit Verweis auf dieses Problem veröffentlichen wir einen CPC-Klassifikationsdatensatz mit den vollständigen Patenttexten.

Auch wenn es sich bei der CPC-/IPC-Klassifikation um ein hierarchisches Klassifikationsproblem mit mehreren Klassen pro Dokument handelt, berücksichtigen die meisten früheren Arbeiten bei der Gestaltung der Modellarchitektur nicht die hierarchische Taxonomie und sagen Labels auch häufig nur für eine einzelne Ebene vorher. Mit unserer speichereffizienten Modellarchitektur leisten wir einen wesentlichen Beitrag. Diese nutzt für mehrere Klassifikations-Heads — einen für jede Klasse in der Taxonomie — ein einziges Transformer-basiertes Sprachmodell und implementiert hierarchische Beziehungen in der Modellarchitektur. Wir zeigen, dass das vorgeschlagene Verfahren kontinuierlich besser als existierende Referenzysteme abschneidet, vor allem bei selteneren Klassen.

Unsere Analyse zeigt, dass die Sätze und Abstracts oft doppelt vorhanden sind, was auf die Redundanz der Patenttexte zurückzuführen ist. Darüber hinaus sind Transformer-basierte Modelle, welche Kontextgrößen von 512 oder 4.096 haben, unzureichend für Patente, die durchschnittlich 12.500 Tokens enthalten. Dies hat uns motiviert, mit einem Verfahren für eine bessere Dokumentendarstellung einen wichtigen Beitrag zu leisten, indem wir Textembeddings aus gekürzten Abschnitten erstellen und diese mittels Vektoraddition kombinieren. Zudem schlagen wir ein Satzpriorisierungssystem vor, das die informativsten Sätze für die Darstellung von Dokumenten auswählt. Wir zeigen, dass die extraktiven Zusammenfassungstechniken effektiv dabei sind, informative Sätze für die Patentklassifizierung zu identifizieren.

Anders als bei der CPC-/IPC-Klassifikation sind bei der PLS die CPC-/IPC-Klassen bei der Inferenz bekannt. Ein wichtiger Beitrag für die bessere Darstellung von Dokumenten ist die Kombination von CPC-/IPC-Klassen mit Patenttext. So prognostizieren wir PLS-orientierte Kategorien, die oft andere Konzepte als die CPC-/IPC-Klassen darstellen. Um die breitere Anwendbarkeit des vorgeschlagenen Verfahrens zu demonstrieren, setzen wir es bei einer ähnlichen Aufgabe ein: für die Klassifikation von Forschungspublikationen in Zielkategorien unter Verwendung von Text und vom Autor bereitgestellten Schlüsselwörtern als Eingabe.

Acknowledgment

During my enriching journey pursuing my Ph.D. at Heidelberg University and the Bosch Center for Artificial Intelligence, I found immense pleasure and invaluable learning experiences. Foremost, my deepest gratitude goes to my supervising professor, Michael Gertz. Under his mentorship, I delved deeper into the realm of natural language processing and gained insights into the practical usability and application of my research. Professor Gertz provided timely and crucial feedback, even amid the busiest periods, shaping my research and guiding its trajectory. Our insightful discussions enabled me to critically analyze my work, laying the foundation for the future course of action.

I extend equal thanks to Jannik Strötgen and Annemarie Friedrich, my advisors at Bosch, who played a pivotal role in my academic growth. Their guidance nurtured my understanding of the research process, helped identify key research gaps, and enhanced my scientific writing. Our weekly meetings and asynchronous communications were consistently fruitful, and I appreciate the time and effort they invested in shaping my research.

A special acknowledgment goes to the fellow PhDs of the BCAI-AIR3 team, including Hendrik Schuff, Stefan Grünewald, Sophie Henning, Youmna Ismaeil, Mingyang Wang, and Timo Schrader. Our stimulating discussions during lunch and coffee breaks enriched my research experience. Their contribution extended beyond discussions, as they also provided valuable proofreading and feedback on the final drafts of my thesis. I am equally grateful to Clint Sebastian and David Schoenleber from BCAI-AII for their feedback and support in proofreading. I want to thank the BCAI-AIR3 team members, including Lukas Lange, Heike Adel, Dragan Milchevski, Daria Stepanova, Trung-Kien Tran, Mohamed Gad-Elrab, and Evgeny Kharlamov, who provided diverse perspectives and constructive feedback on paper drafts and conference presentations, which were instrumental in refining my research.

I appreciate the Ph.D. students from the Data Science Group at Heidelberg University: John Ziegler, Satya Almasian, and Dennis Aumillier. Our discussions during my visit were enlightening, and your responsiveness to my queries was invaluable.

I sincerely thank my coauthors, Mark Giereth, Fryderyk Mantiuk, and Tim Tarsi, for their substantial contributions in improving our research papers. I also thank Patrick Fievet from the World Intellectual Property Organization (WIPO) for his assistance on WIPO datasets. Special recognition goes to Ulrich Klinger from Bosch for his dedicated effort in the labeling process for the Injection Valve dataset and to Axel Grzesik, Christian Höppler, and Franz Grzeschniok for their support in the patent filing and code release process within Bosch.

Beyond the academic realm, I owe a debt of gratitude to my family and friends. My parents' unwavering encouragement and support have been a constant throughout my pursuits. Finally, I extend my deepest thanks to my wife, Sunita, whose continuous support was a pillar of strength, especially during the challenging times of the COVID-19 pandemic, keeping me motivated and focused on my work.

Contents

Chapter 1

Introduction

Millions of digitized documents are added to document collections every year, making manual searches infeasible. However, the availability of open-source and proprietary search systems allows users to express their informational needs as search queries and retrieve relevant documents. Web search platforms such as Google Search and Bing index web pages and provide access through a search interface.[1] In addition to webpage searches, specialized search platforms also exist for specific document collections. Patent archives are one such document collection.

Patents are business critical because, if granted by the examination office, a patent confers exclusive rights on the inventors and restricts competitors from using the invention for monetary gain. Intellectual property infringement by a competitor also invites legal action. Analyzing stock and patent infringement data for the United States from 1984 to 1999, Bessen and Meurer (2008) found that by the late 1990s, the cost of infringement was over $16 billion per year, and the average legal cost of an infringement was half a million dollars. Due to their high monetary value, business organizations safeguard their patent portfolios and keep track of inventions registered by competitors, a task that is enabled by the application of patent search and analysis methods. Thus, patent search and analysis is not only a challenging research area but also highly relevant in practical terms due to the importance of this field for business.

Several open-source and proprietary systems are available for patent search, for example, Google Patent Search[2], Patbase[3], which allows keyword-based search through its primary user interface. However, it is infeasible to formulate a search query in specific search scenarios. For instance, when searching and categorizing documents relevant to a certain technical category during the patent landscape study process, a domain expert will often use keyword-based search queries to identify and categorize patents. For broad technical categories, e.g., "methods for understanding and encoding human natural language",

[1]https://www.google.com/ http://www.bing.com [last accessed December 10, 2023]
[2]https://patents.google.com/ [last accessed December 10, 2023]
[3]https://www.patbase.com/ [last accessed December 10, 2023]

it is infeasible for a domain expert to develop an extensive list of keywords (Giczy et al., 2022). Furthermore, the search query must be updated as the terminology used in the domain evolves. Moreover, using an extensive list of keywords can result in a high proportion of false positives in the result set.

In such scenarios, a document classification system could be helpful (Manning et al., 2008, page 253). A typical document classification system incorporates a model that learns the classification task with a few labeled examples as input. During inference, it takes a document as input and maps it to a set of labels representing pre-defined domain concepts or topics. Later, these labels can be used for retrieval to identify and filter documents relevant to a user's information needs. In addition to facilitating document searches, a labeled document collection also assists in analysis. For example, Mahlia et al. (2020) study technological trends in biodiesel production techniques by categorizing patents into five categories, such as catalysts and pre-treatment methods, and then performing an analysis. For many years, patent categorization was performed manually, requiring a massive effort on the part of domain experts, but automatic document classification systems have now been adopted to replace or support manual patent labeling processes (Larkey, 1998; Lee and Hsiang, 2019; Zaheer et al., 2020; Choi et al., 2022; Yücesoy Kahraman et al., 2023).

Patents exhibit some peculiar characteristics that must be considered and leveraged when developing patent classification techniques. For instance, patents vary widely in document length compared to other document types, such as research publications and newspapers. For example, US7296968B2[4] is described in only four pages whereas US1022496-1B2[5] comprises more than 490 pages. Furthermore, the hierarchical structure of existing patent taxonomies can be leveraged to develop automatic classification models that learn to predict concepts at different levels of granularity (see Figure 2.3, page 15). In this thesis, we analyze patents and describe some key characteristics and challenges associated with patent analysis and search. Based on our analysis, we propose novel classification techniques and evaluate them on classification tasks in the context of crucial patent use cases.

In Section 1.1, we discuss further motivations for this thesis by describing key patent use cases that involve classification tasks. In Section 1.2, we provide details on the key challenges and contributions in this area of research and outline the thesis. Finally, in Section 1.3, we provide information on research publications, datasets, and code repositories published in connection with this thesis.

1.1 Motivating Patent Classification

An invention is a novel and non-obvious idea that can be realized as a process, system, or design. To obtain exclusive rights over an invention, an inventor submits a patent application, which is usually drafted by a patent attorney, to the examination office. A patent ap-

[4]https://patents.google.com/patent/US7296968B2 [last accessed December 10, 2023]
[5]https://patents.google.com/patent/US10224961B2 [last accessed December 10, 2023]

plication comprises four main textual fields: title, abstract, claims, and description. The title indicates the high-level topic of the invention, while the abstract provides a concise summary. The claims field specifies the invented process or system, and the description section elaborates on the details of the invention and the background work that led to it.

In addition to these text fields, a set of labels is associated with a patent where the labels are taken from a pre-defined patent taxonomy. Over the years, patent examination offices have devised patent taxonomies that contain important domain concepts.[6] The International Patent Classification (IPC)[7] and the Cooperative Patent Classification (CPC)[8] are the two predominant patent taxonomies. These taxonomies contain labels representing key domain concepts, which are arranged in a hierarchical structure. Further, a label description is associated with each label and describes the concept. Depending on their hierarchical level, CPC labels are arranged in five hierarchical categories: "section", "class", "subclass", "group", and "subgroup". The labels at the first, second, third, and fourth levels are categorized into "section", "class", "subclass", and "group", respectively. The labels in the fifth level and below belong to the "subgroup" category. Although the patent taxonomies define essential concepts, a domain concept may not have a one-to-one mapping in the label set. For instance, the "computers and office machinery" category might overlap with multiple CPC labels corresponding to typewriters, computing devices, storage devices, and printers. In such cases, domain experts may define a novel domain-specific taxonomy. For example, Inaba and Squicciarini (2017) propose a novel taxonomy of concepts relevant to the Information and Communication Technologies field.

According to Alberts et al. (2011), patent search and analysis processes are particularly relevant to two key user groups. The first is made up of professionals in patent examination offices, such as patent examiners and analysts, who primarily use patent search tools to classify patent applications into a set of CPC/IPC labels and identify relevant prior art. The second comprises professionals within business organizations who use patent search and analysis processes for various applications, such as studying research and business trends and identifying potential patent infringements by their competitors.

In this thesis, we define conceptual models, release datasets, perform analyses, and propose novel document classification techniques that will benefit users in business-critical patent use cases. Below, we discuss key patent use cases and how classification techniques can reduce manual effort in each case.

- *CPC/IPC classification:* On submission to the examination office, a patent is assigned a set of labels taken from the CPC/IPC taxonomy, which facilitates the assignment of the application to the appropriate department within the examination office (Krier and Zaccà, 2002). The assigned CPC/IPC labels also assist in search and analysis tasks. For example, when performing a search by using a query to

[6]https://www.epo.org/searching-for-patents/helpful-resources/first-time-here/classification [last accessed December 10, 2023]

[7]https://www.wipo.int/classifications/ipc/en/ [last accessed December 10, 2023]

[8]https://www.uspto.gov/web/patents/classification/cpc/html/cpc.html [last accessed December 10, 2023]

identify relevant patents, the CPC/IPC labels can be used as an additional filter to reduce the number of false positives returned by the search. The CPC/IPC classification task was performed manually for many years before the introduction of machine learning-based automated classification systems (Larkey, 1999).

- *Patent landscape study:* A Patent Landscape Study (PLS) provides a snapshot of patents relevant to a specific field. Typically, the findings of a PLS are published as a report providing insights that help investors, corporations, research institutions, and policymakers make informed decisions.[9] A typical PLS consists of three steps: search, classification, and analysis (Abood and Feltenberger, 2018). In the first step, a domain expert defines the scope of the PLS, formulates a search query, and identifies relevant patents using patent search systems. In the second step, the relevant patents are categorized into PLS-oriented labels representing concepts such as technical fields, products, and processes. Finally, the labeled dataset is analyzed to draw out essential insights, which are then documented in a PLS report. A patent classification system is utilized in the first and second steps of a PLS. In the first step, a binary classifier can help classify candidate patents into relevant and non-relevant categories. In the second step, a multi-label classifier can help to predict a set of PLS-oriented labels for a patent.

- *Patent alert:* Each week, thousands of new patents are published by patent examination offices across the globe, making it challenging for inventors to identify relevant patents.[10] A patent alert system can be helpful in this regard. In a typical patent alert generation system, a user registers for periodic patent updates, which contain a subset of newly registered patents shortlisted based on a user-provided search query [11]. Domain experts generally aim to ensure that they are notified about critical inventions. They make extensive use of keywords and CPC/IPC labels as search queries to maximize the retrieval of relevant patents. However, a query with a broader scope may result in many non-relevant patents being mistakenly identified as relevant. In this case, a patent classification system trained with a few labeled examples can help reduce the number of false positives returned by the search.

- *Prior art search:* In a patent examination office, a patent is granted after the proposed invention is examined against novelty and non-obviousness as two of the main criteria (Franzosi, 2000). To assess this, a patent examiner evaluates the invention by comparing it with prior art, which includes patent and non-patent documents such as research publications, technical reports, and blog posts. This task bears similarity to the document-to-document retrieval task, and a document representation devised and trained for CPC/IPC classification might find an application in a document-to-document retrieval task (Shalaby and Zadrozny, 2019).

[9]https://www.ipcheckups.com/patent-landscape-analysis-overview/ [last accessed December 10, 2023]

[10]https://bulkdata.uspto.gov/ [last accessed December 10, 2023]

[11]https://minesoft.com/pairalerts/ [last accessed December 10, 2023]

The patent classification models developed in this thesis primarily target and evaluate the CPC/IPC classification and patent landscape study use cases. However, the document representation methods and classification models proposed in this thesis may also be valuable in patent alert and prior art search scenarios.

1.2 Outline, Challenges, and Contributions

In the previous section, we briefly discussed business-critical patent use cases involving classification tasks that can be addressed using classification techniques. Here, we detail the main challenges addressed in each chapter and the contributions made by this thesis.

Chapter 2: Background. In Chapter 2, we provide the context for our work by discussing key patent classification use cases and tasks in detail. Further, we describe the text- and graph-based representation methods and the neural and non-neural classification models that serve as building blocks for the document classification techniques described in the subsequent chapters.

Chapter 3: Conceptual Model for Document Classification. The document classification model is incorporated into a business solution as one of the components. A major engineering challenge is to adapt the classification pipeline for new use cases and incorporate novel classification techniques. To achieve this, we propose a conceptual model for a classification pipeline that provides a unified view to stakeholders and allows them to make crucial decisions regarding system design.

Chapter 4: Datasets and Analyses. One of the most significant challenges involved in the development and evaluation of patent classification models is the lack of labeled datasets. To the best of our knowledge, this thesis presents the first benchmark for patent landscape study classification using natural language processing (NLP) technology. As a significant contribution of this thesis, we *curate and release three datasets from two diverse domains* (mechanical and biochemical) and use them to evaluate the classification models in the context of patent landscape study (Pujari et al., 2022b). Furthermore, most of the previous work on CPC/IPC classification only utilizes titles, abstracts, and claims as patent fields, omitting the more detailed description section because of the efficiency reasons (Li et al., 2018a; Lee and Hsiang, 2019; Zaheer et al., 2020). To facilitate research on long-text representation methods in general and patent representation with full texts of patents in particular, we *release a CPC classification dataset in which each instance contains the full text of patent*, allowing the evaluation of document representation methods in Chapter 6 (Pujari et al., 2022a). As the characteristics of the dataset influence the design of the document representation methods and help us interpret the model outcomes, we perform a crucial analysis of duplicate texts. We *find that abstracts are often reused across patents*. In a few instances, we find that the *sentences are also duplicated* within and across patents. The analysis of duplicate text detailed in this chapter constitutes a further contribution and motivates the document representation techniques described in Chapter 6, which utilize a limited set of the most informative text elements.

Chapter 5: CPC Classification using Transformers. Although the CPC/IPC is a hierarchical taxonomy, most previous works have addressed it using flat classification approaches that only predict leaf-level labels. We find that a recent hierarchical non-neural approach, namely TwistBytes (Benites, 2019), performs comparably to a flat neural approach based on the ULMFiT contextual language model (Hepburn, 2018) evaluated on the hierarchical classification task (Remus et al., 2019). TwistBytes is a local classifier per node (LCN) based approach that trains multiple classifiers, one for each label, and uses hierarchical links during prediction. Further, transformer-based classifiers perform better in various NLP tasks (Craswell et al., 2020). However, it is infeasible to train a neural LCN classifier with a transformer-based language model for text representation due to memory constraints.

To overcome these challenges, a significant contribution of this thesis is the development of a novel memory-efficient model architecture, namely *Transformer-based Multi-task Model (TMM)*. The TMM consists of multiple classification heads, one for each label, which predict whether a label applies to an input representation. Further, the classification heads share a common transformer-based language model for text representation. The proposed model outperforms multiple neural (Li et al., 2018a; Lee and Hsiang, 2019; Huang et al., 2019) and non-neural baselines (Benites, 2019). In addition, we propose a *hierarchical variant called the Transformer-based Hierarchical Multi-task Model (THMM)*. The THMM leverages hierarchical taxonomy links to transfer representation from the head corresponding to a parent label to the heads corresponding to its children. Based on a comprehensive analysis, we demonstrate that the hierarchical links in the THMM improve performance—particularly at the third level, which contains many infrequent labels (Pujari et al., 2021a).

Pre-trained models perform better on a task if the text used for pre-training has higher proximity with the target domain (Sung et al., 2019). Being pre-trained on scientific text, the vocabulary for SciBERT (Beltagy et al., 2019) has higher overlap with patent terminology than the vocabulary of Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), which was pre-trained on English Wikipedia and BookCorpus (Zhu et al., 2015). We achieved better performance with SciBERT and, for this reason, used it for text representation in our experiments.

Chapter 6: Efficient Neural Full-Text Patent Representations. Leveraging the full texts of patents poses computational challenges for text representation methods, especially transformer-based models that take 512 or 4096 tokens as input (Devlin et al., 2019; Beltagy et al., 2019, 2020; Zaheer et al., 2020). In our analysis in Chapter 4, we discover that, on average, patents contain around 12.5k tokens (see Figure 4.5, page 61). However, BERT-based models allow a maximum sequence length of 512 tokens, and the maximum sequence length for models employing sparse-attention mechanisms, such as Longformer

(Beltagy et al., 2020), BigBird (Zaheer et al., 2020), is 4096 tokens. In addition, our analysis of duplicate text in Chapter 4 reveals the redundant nature of patent texts. Furthermore, in their evaluation of long text representation methods, Park et al. (2022) found that a classifier based on Longformer is inefficient, and for some datasets, it performs worse than a BERT-based baseline.

Considering these factors, we propose an efficient data-driven approach that identifies the subset of the most informative text elements (e.g., fields, sentences, and tokens) for patent representation. Our approach is based on the hypothesis that the redundancy of the information within a patent means that it can be represented using a subset of the most informative text elements, such that adding more elements does not enhance classification accuracy. For efficiency reasons, we continue with the SciBERT model fine-tuned in Chapter 5 and generate embeddings for texts corresponding to the patent fields and sentences without further fine-tuning the SciBERT parameters. As a significant contribution of Chapter 6, we propose an approach that combines the embeddings from section texts using vector summation, which achieves the best performance in the CPC classification task (Pujari et al., 2022a). As a further contribution, we propose a novel sentence ranker that computes a score based on the similarity of a sentence to the descriptions of CPC labels, the position of a sentence within the patent section, and the relative importance of the corresponding patent section text when evaluated with the classification task. With comprehensive evaluations, we show that extractive summarization is effective in selecting a subset of the most informative texts for neural representation in the context of patent classification.

Chapter 7: Neural Representations for Patent Landscape Study. The classification task in the context of PLS differs from CPC/IPC classification as it is performed over patent applications and grants. Therefore, the CPC/IPC labels are known during inference. The CPC/IPC labels represent crucial domain concepts, and incorporating them into the document representation may improve classification performance. As a significant contribution of Chapter 7, we propose a neural computational model that combines patent text and CPC/IPC label information and performs robustly across heterogeneous patent landscape study datasets when compared to both neural and non-neural baselines (Pujari et al., 2022b). Furthermore, we showcase the applicability of the proposed techniques to classification settings other than patents through our participation in Task 5 of the BioCreative VII challenge, which focuses on predicting COVID-19-related class labels for PubMed articles (Chen et al., 2021b). This demonstrates that despite our focus on patents, our proposed methods are robust and can be applied in other settings with similar characteristics (Pujari et al., 2021b).

Chapter 8: Conclusions and Future Work. In Chapter 8, we discuss our main findings, the research outcomes, and their implications for the field of information retrieval and natural language processing. We discuss the significance of our work in advancing state-of-the-art patent classification methods and highlight its limitations, acknowledging areas where further improvements can be made.

1.3 Research Publications, Datasets, and Code Repositories

Parts of the research described in this thesis have been published in peer-reviewed venues, accompanied by datasets and code to facilitate the reproducibility of experiments. In the following, we provide details on the research publications, code repositories, and released datasets produced in connection with this thesis.

Publications. The work in this thesis primarily relates to the following peer-reviewed articles (ordered by publication date).

[1] **Subhash Chandra Pujari**, Annemarie Friedrich, and Jannik Strötgen. 2021. A Multi-task Approach to Neural Multi-label Hierarchical Patent Classification Using Transformers. In *Proceedings of the 43rd European Conference on Information Retrieval (ECIR '21).*

The classification model architecture and evaluations in Chapter 5 are based on this work.

[2] **Subhash Chandra Pujari**, Tim Tarsi, Jannik Strötgen, Annemarie Friedrich. 2021. Team RobertNLP at BioCreative VII LitCovid Track: Neural Document Classification Using SciBERT. In *Proceedings of the BioCreative VII Challenge Evaluation Workshop.*

The out-of-domain evaluation performed in Chapter 7, which demonstrates the applicability of our methods to domains other than patents, is based on this work. The author supervised Tim Tarsi during this work, and Tim contributed to data cleaning and analysis. The final evaluations and analyses mentioned in the paper were performed by the author.

[3] **Subhash Chandra Pujari**, Fryderyk Mantiuk, Mark Giereth, Jannik Strötgen, and Annemarie Friedrich. 2022. Evaluating Neural Multi-Field Document Representations for Patent Classification. In *Proceedings of the 12th International Workshop on Bibliometric-enhanced Information Retrieval (BIR '22) co-located with 44th European Conference on Information Retrieval (ECIR '22).*

Part of this research was conducted in the context of Fryderyk Mantiuk's bachelors thesis, which was co-supervised by the author. However, the final experiments, evaluations, and additional analyses mentioned in the paper were performed by the author. The experiments and results from this work are mentioned in Chapter 6.

[4] **Subhash Chandra Pujari**, Jannik Strötgen, Mark Giereth, Michael Gertz, and Annemarie Friedrich. 2022. Three Real-World Datasets and Neural Computational Models for Classification Tasks in Patent Landscaping. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, (EMNLP '22).*

This work presents the datasets (discussed in Chapter 4) and neural network model architecture (discussed in Chapter 7) used to address the classification task in the PLS context. The injection valve dataset was created within Robert Bosch GmbH[12] and is annotated by Ulrich Klinger, an expert in injection valve-related technologies. Mark Giereth contributed to the initial data cleansing process.

Published Code. Implementations of the proposed classification techniques, evaluations, and analyses are publicly available on GitHub under open source software license:

- Implementation of TMM and THMM models (see Section 5.3).
  URL: *https://github.com/boschresearch/hierarchical_patent_classification_ecir2021* [last accessed December 10, 2023]
- Implementation of document representation to evaluate field text for patent representation (see Section 6.3).
  URL: https://github.com/boschresearch/multifield_patent_classification_bir2022 [last accessed December 10, 2023]
- Implementation of neural network model for classification in the PLS context (see Section 7.3).
  URL: *https://github.com/boschresearch/pls_benchmark_emnlp2022* [last accessed December 10, 2023]

Released Datasets. We provide the following links to the USPTO-70k and PLS classification datasets.

- We release the USPTO-70k dataset for the evaluation of CPC classification techniques. Unlike previous datasets (Li et al., 2018a; Abdelgawad et al., 2019), each instance in the dataset comprises the full text of the patent. Thus, the released dataset is crucial for research on long-text classification methods. The USPTO-70k dataset is described in Chapter 4 and is used for evaluation in Chapter 5 and Chapter 6.
  URL: *https://zenodo.org/record/6992298* [last accessed December 10, 2023]
- We release the three patent landscape study datasets described in Chapter 4 and used for evaluation in Chapter 7.
  URL: *https://github.com/boschresearch/pls_benchmark_emnlp2022/tree/main/patent_ls* [last accessed December 10, 2023]

[12] https://www.bosch.de/ [last accessed December 10, 2023]

Chapter 2

Background

In this thesis, we propose novel document representation methods and classification model architectures that build on the key concepts defined in the current chapter. In this chapter, we provide a brief overview of classification models and representation techniques and mention key references, particularly those in the field of patent classification. Subsequent chapters provide detailed discussions of work based on the key concepts used in the respective chapters.

In Section 2.1, we provide a brief overview of the patent examination process, key patent taxonomies, and the structure and content of a typical patent. A description of different patent analysis use cases and associated tasks is provided in Section 2.2. In Section 2.3, we provide an overview of the process of selecting, initializing, training, and validating a document classification model. The document representation techniques proposed in Chapters 6 and 7 build on the text and graph-based representation methods described in Sections 2.4 and 2.5, respectively. The classification model architecture introduced in Chapter 5 is inspired by the neural and non-neural classification model architectures discussed in Section 2.6. In Section 2.7, we describe the evaluation metrics used to evaluate the performance of patent classifiers in hierarchical and non-hierarchical patent classification tasks in Chapters 5, 6, and 7.

2.1 Patent

In this section, we describe a typical patent examination process followed by a description of the content and structure of a patent document, which is accompanied by details about patent taxonomies.

2.1.1 Examination Process

During the examination process, an invention is documented in a patent application submitted to an examination office, where it is evaluated against different criteria, resulting

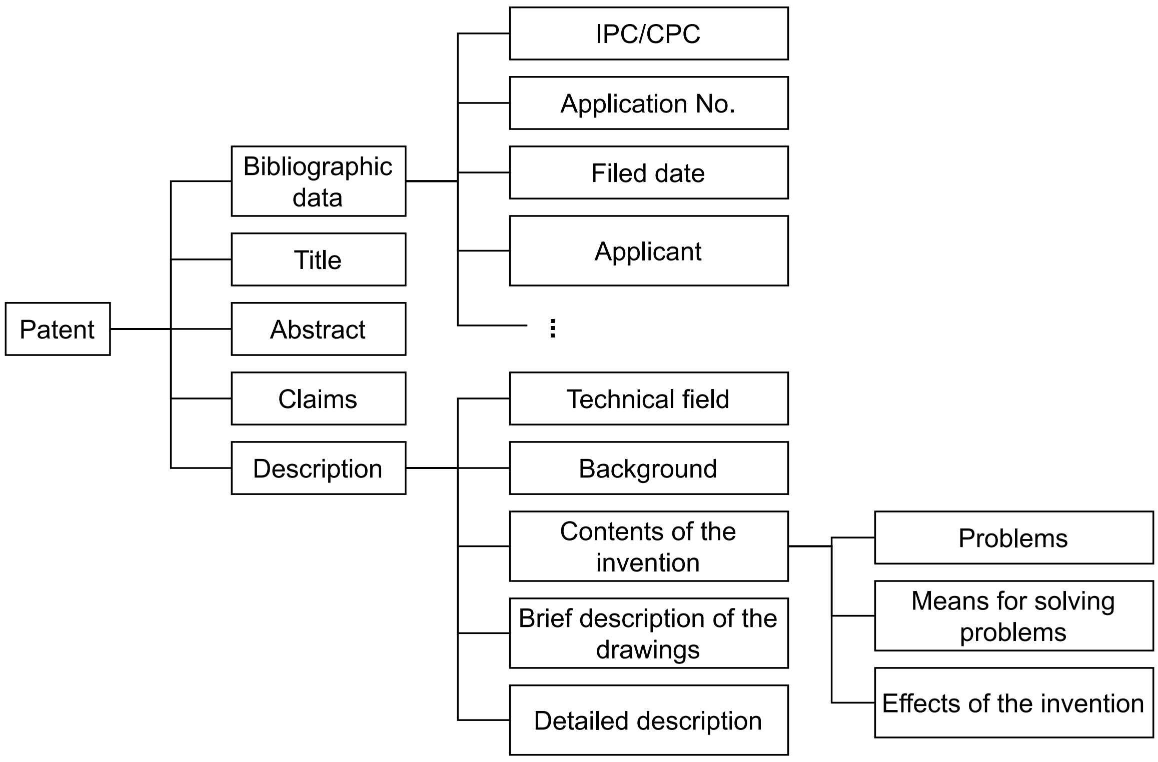


Figure 2.1: The structure of a typical patent based on Lim and Kwon (2016).

in acceptance or rejection. Acceptance results in a patent grant, allowing the inventors to make commercial use of the invention within the jurisdiction of the patent office. Prominent patent offices include the United States Patent and Trademark Office (USPTO)[1], the European Patent Office (EPO)[2], the China National Intellectual Property Administration (CNIPA)[3], and the Japanese Patent Office (JPO).[4] Since a patent grant is valid only within the relevant patent office's geographical boundaries, separate applications for the same invention are filed in patent examination offices across the globe to ensure broader applicability (Carsten Fink and Zhou, 2016).

Here, we summarize a typical patent examination process within the USPTO based on the details provided by Marco et al. (2019). On submission to an examination office, a patent is labeled with a set of labels taken from a pre-defined taxonomy so that it can be forwarded to the correct department and examiner within the examination office. The first step is to evaluate the application for completeness. Next, an examiner evaluates whether the application contains claims targeting a single invention or whether they target multiple inventions. Further, the novelty and non-obviousness of the application are evaluated—these are the two main criteria based on which an application might be accepted or rejected. Even if the application is rejected, there remains the possibility of re-submission once the objections raised by the patent examination office have been addressed.

[1]https://www.uspto.gov/ [last accessed December 10, 2023]
[2]https://www.epo.org/ [last accessed December 10, 2023]
[3]https://english.cnipa.gov.cn/ [last accessed December 10, 2023]
[4]https://www.jpo.go.jp/e/ [last accessed December 10, 2023]

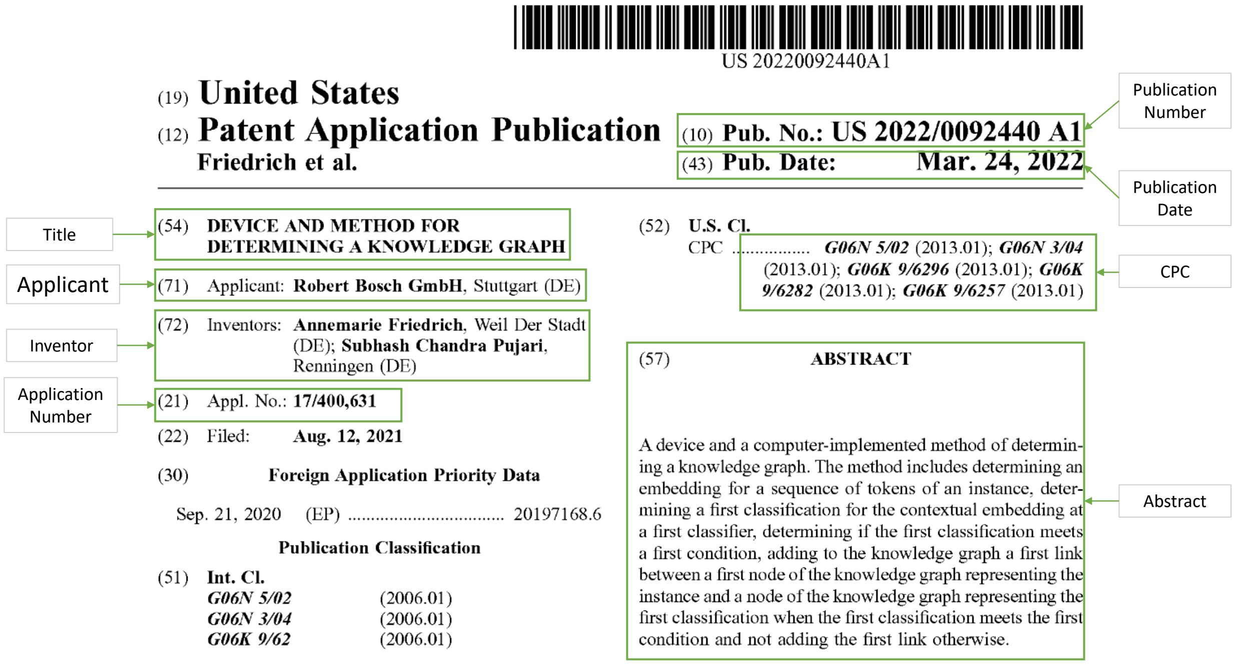
2.1.2 Patent Fields

The structure of a typical patent document is shown in Figure 2.1. A patent contains the specifications of an invention described across different textual fields at various levels of granularity. A typical patent template is comprised of four textual fields (Lim and Kwon, 2016), which are defined as follows:

- The *title* provides high-level information on the topic of an invention. Unlike research publications, the titles of patents are often duplicated, such that the same title might appear in several different patents. In Section 4.6, our analysis shows that some titles, such as "*tire*" and "*bottle*", appear across thousands of patents.
- The *abstract* concisely summarizes an invention and allows the reader to quickly assess its relevancy. However, like titles, patent abstracts are not always unique: the same abstract might be associated with multiple patents (see Section 4.6). As abstracts do not have legal significance, they are written incautiously, and therefore, information from additional fields is crucial for the comparison of nuances.
- The *claims* section contains a set of claims which constitute a detailed specification of an invention. The claims section is the only patent section with legal significance and can be challenged with litigation in the case of infringement. Due to these legal implications, claims are often written using non-standard terminology, with the main aim being to broaden the scope of an invention. In a commonly observed pattern, disclosure of a system in a patent might describe its functionality or usage, and for this reason, ambiguous and abstract terminology may be used. For example, US20050089604A1[5] for the invention of an ice cream chip refers to it as "*An edible crisp unitary pastry having a double-curvature and having a planar longitudinal axis and a planar latitude axis perpendicular to the longitudinal axis*". Such use of non-standard terminology makes it challenging to determine whether two terms refer to the same invention. Although the claims section contains the complete specification of an invention, the use of non-standard terminology makes it difficult to use the associated information and compare two similar inventions. For this reason, claims are often interpreted based on the description field rather than just the claims field.
- In addition to claims, the *description* field provides detailed information about the invention and its background. The description field consists of additional sub-fields: technical field, background, brief summary, drawing description, and detailed description. Among these different sub-fields, the detailed description field is the longest and consists of more than 9k tokens on average (see Figure 4.5d, page 61).

In addition to the content fields, metadata information is associated with a patent, primarily so that records can be managed within patent examination offices. Details on metadata fields can be found in the documentation for preparation of a patent application as

[5]https://patents.google.com/patent/US20050089604A1 [last accessed December 10, 2023]



Publication Number
Publication Date
Title
Applicant
CPC
Inventor
Application Number
Abstract

US 20220092440A1

(19) **United States**
(12) **Patent Application Publication**
Friedrich et al.

(10) **Pub. No.: US 2022/0092440 A1**
(43) **Pub. Date: Mar. 24, 2022**

(54) **DEVICE AND METHOD FOR DETERMINING A KNOWLEDGE GRAPH**

(71) Applicant: **Robert Bosch GmbH**, Stuttgart (DE)

(72) Inventors: **Annemarie Friedrich**, Weil Der Stadt (DE); **Subhash Chandra Pujari**, Renningen (DE)

(21) Appl. No.: **17/400,631**

(22) Filed: **Aug. 12, 2021**

(30) **Foreign Application Priority Data**

Sep. 21, 2020 (EP) 20197168.6

Publication Classification

(51) **Int. Cl.**

G06N 5/02	(2006.01)
G06N 3/04	(2006.01)
G06K 9/62	(2006.01)

(52) **U.S. Cl.**
CPC ***G06N 5/02*** (2013.01); ***G06N 3/04*** (2013.01); ***G06K 9/6296*** (2013.01); ***G06K 9/6282*** (2013.01); ***G06K 9/6257*** (2013.01)

(57) **ABSTRACT**

A device and a computer-implemented method of determining a knowledge graph. The method includes determining an embedding for a sequence of tokens of an instance, determining a first classification for the contextual embedding at a first classifier, determining if the first classification meets a first condition, adding to the knowledge graph a first link between a first node of the knowledge graph representing the instance and a node of the knowledge graph representing the first classification when the first classification meets the first condition and not adding the first link otherwise.

Figure 2.2: The bibliographic information for an example USPTO patent with the publication number "US 2022/0092440 A1". The document in the figure is the published version of a patent application under review.

provided by the European Patent Office.[6] The major metadata fields within a patent are marked in Figure 2.2 with "US 2022/0092440 A1"[7] application as example and are described as follows:

- *Applicant and inventors*: The *applicant* field specifies the name of the organization that is filing an invention, whereas the names of the individual contributors are mentioned as *inventors*. An organization or an inventor might be referred to with different names in applications both within and across patent offices, thus leading to the problem of name ambiguity (Haak et al., 2012).

- *Application date and publication date*: The date on which a patent application is filed at the examination office is referred to as the *application date*. In contrast, the *publication date* refers to the date on which the examination office publishes an application for public access.

- *Publication number*: After reviewing a patent application, the examination office assigns it a *publication number* as a unique identifier. A publication number comprises a country code, a document number, and a kind code. The country code indicates the region where a patent will be valid if granted. The document number is a unique identifier for a patent. The kind code indicates the current state of a patent document.

[6]https://www.epo.org/en/legal/guide-epc/2022/ga_c4.html [last accessed December 10, 2023]
[7]https://patents.google.com/patent/US20220092440A1/en [last accessed December 10, 2023]

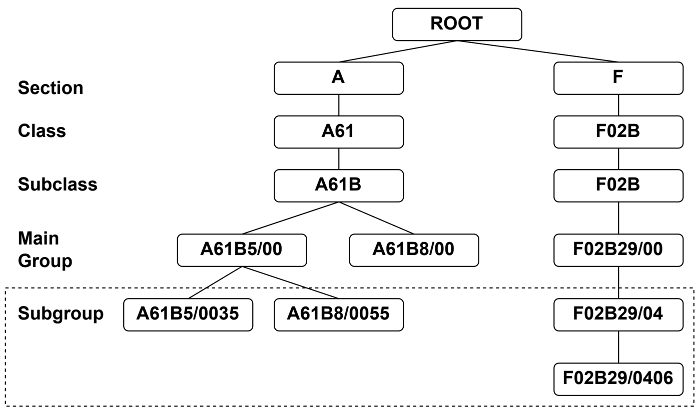


Figure 2.3: An example showing the hierarchical structure of the Cooperative Patent Classification (CPC) scheme.

Kind codes with the prefix “A”, for example, A1 or A2, denote different versions of the patent application, whereas documents with the kind code prefix “B” are patent grants. The publication number shown in Figure 2.2 contains “US” as the country code, “2022/0092440” as the document number, and “A1” as the kind code.

- *Family number*: Related patent applications filed within the same or different examination offices are linked by the same *family number* identifier. The EPO defines two types of patent families: the first is referred to as a simple patent family and contains the same invention, while the second is known as an extended patent family and contains similar inventions.[8]

- *CPC/IPC labels*: The example application shown in Figure 2.2 contains labels from both the Cooperative Patent Classification (CPC)[9] and International Patent Classification (IPC)[10] schemes, both of which are pre-defined patent taxonomies. The patent taxonomies contain labels that represent concepts occurring across patents, which are arranged in a hierarchical structure such that labels at a higher level represent coarse concepts, whereas those at the lower levels represent fine-grained concepts (Figure 2.3). More details on these taxonomies are provided in the following section.

- *Citations*: In addition to the metadata information given on the first page of a patent, a patent is linked to the relevant prior art. Alcácer et al. (2009) found that crucial citations are not mentioned during patent submission and are later added by the examiner during the examination process.

[8]https://www.epo.org/searching-for-patents/helpful-resources/first-time-here/patent-families.html [last accessed December 10, 2023]

[9]https://www.cooperativepatentclassification.org/cpcSchemeAndDefinitions/table [last accessed December 10, 2023]

[10]https://www.wipo.int/classifications/ipc/en/ [last accessed December 10, 2023]

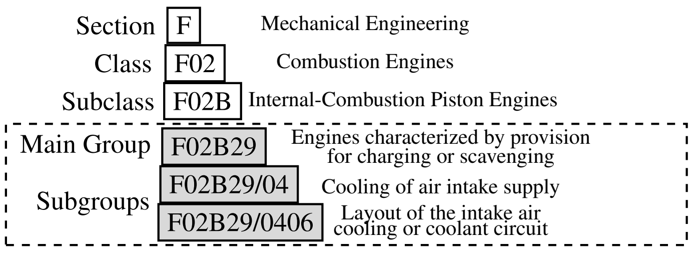

Section
F
Mechanical Engineering
Class
F02
Combustion Engines
Subclass
F02B
Internal-Combustion Piston Engines
Main Group
F02B29
Engines characterized by provision for charging or scavenging
Subgroups
F02B29/04
Cooling of air intake supply
F02B29/0406
Layout of the intake air cooling or coolant circuit

Figure 2.4: An example of the Cooperative Patent Classification (CPC) scheme showing the different hierarchical levels, an example label for each level, and associated label descriptions.

2.1.3 Patent Taxonomies

A patent taxonomy contains labels representing key concepts occurring across inventions so that the most relevant labels can be assigned to each patent; these labels can later be used for search and analysis. Over the years, patent examination offices have devised classification schemes to represent knowledge and organize records within the examination office. Two popular pre-defined patent taxonomies are the International Patent Classification (IPC) and the Cooperative Patent Classification (CPC) taxonomies. The CPC taxonomy was introduced in 2013 and extends the existing label set of the IPC taxonomy with an additional set of fine-grained labels (Degroote and Held, 2018).

The CPC and IPC taxonomies follow hierarchical structures, with labels arranged at different levels. Labels at higher levels represent coarser concepts, while those at lower levels represent finer-grained concepts. Figure 2.3 shows an example of the hierarchical structure of the CPC taxonomy, which contains labels categorized into five groups based on hierarchical level. The first-level group is referred to as “Section” and contains nine labels. The second and third-level groups are referred to as “Class” and “Subclass” and contain 136 and 674 labels, respectively. The fourth-level group is referred to as the “Main Group”, whereas the labels in the fifth level and below belong to the “Subgroup” category. The “Main Group” and “Subgroup” categories contain 9.8k and 250k labels, respectively.

A textual description accompanies each label in the CPC taxonomy. Figure 2.4 gives an example of labels at different levels of granularity and their respective descriptions. Upon analyzing these descriptions, we observe that the description of a child label provides additional information to the concepts defined by the parent label.

While the CPC taxonomy contains more than 260k labels in total, there may be cases in which a domain concept does not have a one-to-one match with a label in the CPC label set. In such scenarios, domain experts may create a new taxonomy with labels representing novel domain concepts. For example, Inaba and Squicciarini (2017) propose a novel taxonomy consisting of 13 subcategories representing concepts relevant to the field of information and communication technologies.

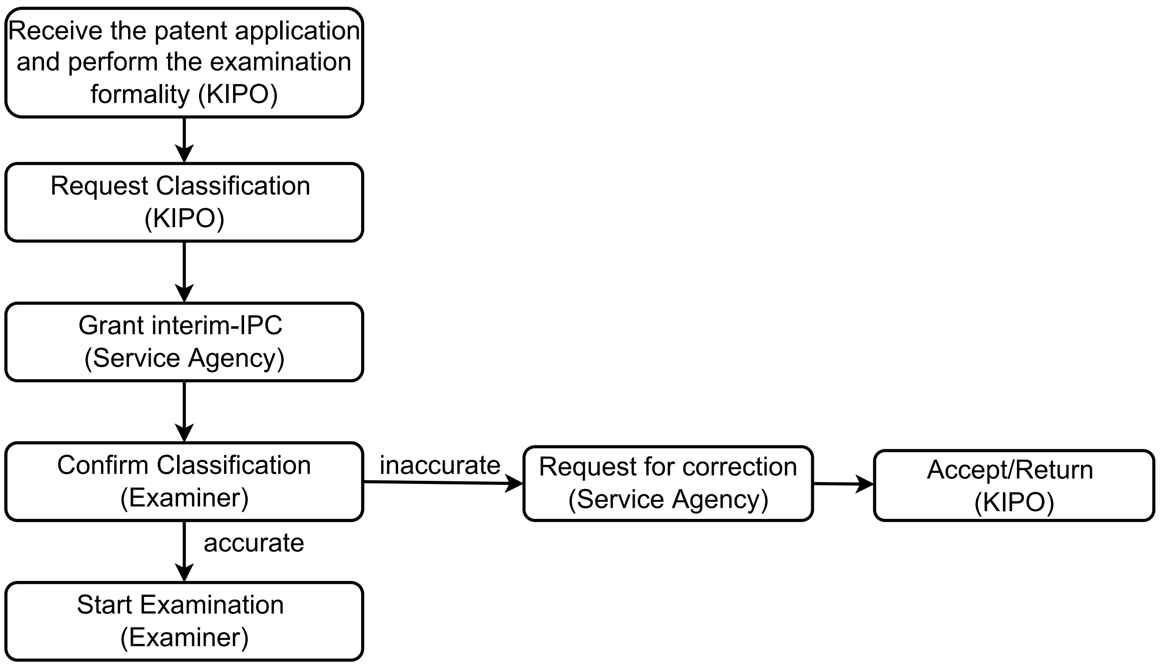

Receive the patent application and perform the examination formality (KIPO)
Request Classification (KIPO)
Grant interim-IPC (Service Agency)
Confirm Classification (Examiner)
inaccurate
Request for correction (Service Agency)
Accept/Return (KIPO)
accurate
Start Examination (Examiner)

Figure 2.5: The IPC labeling process within the Korean Intellectual Property Office (KIPO) as illustrated by Lim and Kwon (2016).

2.2 Use Cases and Tasks

The patent classification problem is encountered in a variety of patent use cases (Krestel et al., 2021), the most important of which can be defined as follows:

CPC/IPC Classification. The CPC/IPC classification use case involves assigning labels from a pre-defined patent taxonomy (Section 2.1.3) to a newly submitted application in the examination office. The typical IPC labeling process is shown in Figure 2.5, which is based on Lim and Kwon (2016). The assigned labels serve two primary purposes: identifying the correct department for evaluating a patent and filtering out irrelevant patents during searches.

The CPC/IPC classification process is predominantly manual, and both the research community and patent examination offices have made continuous efforts to automate it (Fall et al., 2003; Li et al., 2018a; Lee and Hsiang, 2019). Compared to other patent use cases, one factor that facilitates the development of machine learning solutions for CPC/IPC classification is the availability of a large open-source labeled dataset.[11] For this reason, the CLEF-IP (Piroi and Hanbury, 2017) and NTCIR (Lupu et al., 2017) challenges include IPC classification among their tasks.

As discussed in Section 2.1.3, the CPC/IPC labels are distributed across five level-based categories. However, label distribution for the two lower-most label groups is quite skewed. Therefore, most previous works predict labels at the third level of the class hier-

[11] https://patentsview.org/download/data-download-tables [last accessed December 10, 2023]

archy, i.e., the subclass group; this includes the task setup for the CLEF-IP 2011 challenge (Piroi et al., 2011). In addition, CPC/IPC classification has been predominantly addressed using flat classification approaches in which class labels are predicted for a single level of the hierarchy, usually at the third level for CPC/IPC taxonomy (Fall et al., 2003; Li et al., 2018a; Abdelgawad et al., 2019; Lee and Hsiang, 2019; Althammer et al., 2021a). This thesis formulates the CPC/IPC classification problem as a hierarchical multi-label classification task and evaluates classification models' ability to predict labels across levels. The CPC/IPC classification task is defined as follows:

Definition 2.1 (CPC/IPC Classification Task)**.** *Given a hierarchical taxonomy C, and the training data set $\{\langle d^{(i)}, C_d^{(i)}\rangle\}_{i=1}^{n}$, the CPC/IPC classification task estimates a function for mapping a document $d^{(i)}$ to a set of CPC/IPC labels $C_d^{(i)}$, where $C_d^{(i)} \subseteq C$.*

As the CPC/IPC taxonomies are hierarchical and multiple labels from the same level can be assigned to a document, the CPC/IPC classification task is a hierarchical multi-label classification task. Further, as each label at the second or lower level is linked to a parent in the level above, the CPC/IPC classification task imposes the condition that the respective ancestors should also be contained in the output set for each predicted label. Next, we discuss the PLS use case and the associated tasks.

Patent Landscape Study (PLS). For certain business scenarios, an organization must study the intellectual property (IP) landscape within a field (Toole et al., 2020) or in relation to an invention.[12] Knowledge of the patent landscape helps an organization to maintain its strategic position in terms of IP artifacts that assist business decisions, for example, the decision on whether to invest in a certain technology. A Patent Landscape Study (PLS) determines the intellectual property landscape and comprises of three main steps: patent search, categorization, and analysis. As a preliminary step, a domain expert defines the motivation behind the PLS and carves out its scope. Once the scope has been defined, the domain expert identifies patents relevant to the PLS, querying open source or proprietary search systems using keywords and CPC/IPC labels. The patent search step is iterative, as the domain expert often reformulates the search query based on the search results. In the second step, the retrieved documents are labeled with labels from a user-defined taxonomy. It may be that both the search and categorization steps are performed together, in which case a domain expert formulates a label-specific search query, and all the documents retrieved with the search query are categorized using the given label (Toole et al., 2020). Once the patent search and categorization process is complete, the domain expert analyzes the labeled dataset to generate a PLS report that provides crucial business insights.

Previous works on automating the PLS process have focused on the first step, which involves identifying the patents most relevant to the PLS (Abood and Feltenberger, 2018; Choi et al., 2022). In contrast, we focus on the second step of the PLS process, which

[12]https://www.wipo.int/publications/en/details.jsp?id=230&plang=EN [last accessed December 10, 2023]

involves categorizing the retrieved patents into user-defined categories. The patent classification task in the context of PLS differs from the CPC/IPC classification task in that the document corpus for the PLS task comprises patent applications and patent grants labeled with a set of CPC/IPC labels. Therefore, the associated CPC/IPC labels can be incorporated into a document representation, expanding the scope of possible document representation techniques compared to CPC/IPC classification. The PLS classification task can be defined as follows:

Definition 2.2 (PLS Classification Task)**.** *Given the training dataset* $\{\langle d^{(i)}, C_d^{(i)}, L_d^{(i)} \rangle\}_{i=1}^{n}$*, estimate a function for mapping a document,* $d^{(i)}$*, which comes with a set of CPC/IPC labels,* $C_d^{(i)}$*, to a set of PLS-oriented categories,* $L_d^{(i)}$*.*

When performing a PLS, domain experts define a set of PLS-oriented categories based on the relevant field of study. For example, in the case of a PLS performed for the drug atazanavir, the categories included disease names (e.g., “Cancer”, “Hepatitis”) as target categories.[13] PLS classification is a multi-label task since multiple labels can be assigned to one patent. One of the major bottlenecks in the development of PLS classification techniques is the unavailability of a labeled dataset. In Chapter 4, we address this issue by releasing three new PLS datasets. Next, we look into the steps involved in a typical process of training a classifier.

2.3 Model Initialization, Training, Validation, and Selection

In this section, we discuss the key concepts associated with document classification. Given a taxonomy as a set of labels L and a document corpus D, document classification is the task of assigning a set of labels $L_d \subset L$ to a document $d \in D$ represented as $h_\theta : D \rightarrow L$, where L is the set of target labels. The function h_θ is referred to as the document classification model, where θ is a set of *trainable parameters*, learned using labeled examples during the model training process. Apart from the trainable parameters, certain parameters are pre-specified during model initialization and are referred to as *hyperparameters*.

As shown in Figure 2.6, a typical document classification model consists of two components: a document representation method and a classification model (Minaee et al., 2021). A *document representation method* $g(d)$ maps a document d to an n-dimensional feature vector, $\boldsymbol{x}$. The vector $\boldsymbol{x}$ is also referred to as a document representation. A document consists of content and metadata elements at different levels of granularity, which can be used to generate a representation of the document and are referred to as semantic entities by Ingwersen (1994). We refer to them as semantic elements. In this thesis, we propose document representation techniques that build on methods representing semantic elements

[13]https://www.wipo.int/publications/en/details.jsp?id=265&plang=EN [last accessed December 10, 2023]

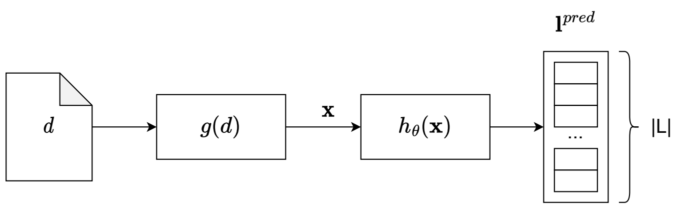
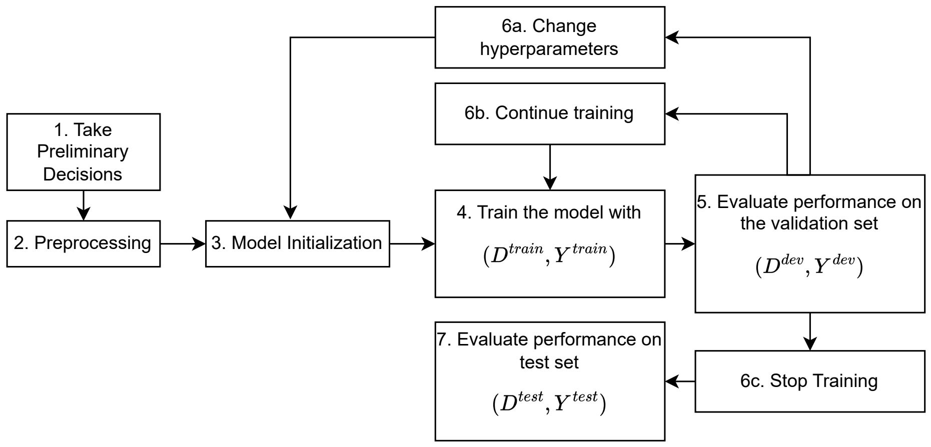


Figure 2.6: The document classification model consists of two components. As the first component, the document representation method $g(d)$ maps a data object d to an n-dimensional feature vector $\boldsymbol{x}$. Next, the classification model $h_\theta(\boldsymbol{x})$ maps $\boldsymbol{x}$ to an output label vector $\boldsymbol{l}^{pred}$.



6a. Change hyperparameters
6b. Continue training
1. Take Preliminary Decisions
2. Preprocessing
3. Model Initialization
4. Train the model with (D^{train}, Y^{train})
5. Evaluate performance on the validation set (D^{dev}, Y^{dev})
7. Evaluate performance on test set (D^{test}, Y^{test})
6c. Stop Training

Figure 2.7: The step-by-step process to initialize, train, validate, and select a classification model, based on Zhang et al. (2021) and Goodfellow et al. (2016).

as a bag-of-words, sequence, or graph, some of which are discussed in Sections 2.4 and 2.5.

A *classification model* takes a document representation as input and generates an $|L|$-dimensional label vector $\boldsymbol{l}^{pred}$ as output (Minaee et al., 2021). The i^{th} value in $\boldsymbol{l}^{pred}$ signifies the *prediction score* of a model for predicting the i^{th} label in the label set L. Thus, the set of predicted labels, $\hat{L}_d$, is determined by selecting all labels with a prediction score exceeding a pre-defined threshold value. In Section 2.6, we provide details on the various classification models that inspired the model architectures described in the later chapters.

The performance of a classification model is evaluated using classification-specific evaluation metrics that take the set of true labels, L_d, and predicted labels, $\hat{L}_d$, as input and compute a score (Minaee et al., 2021). The evaluation metric mimics the real-world scenario in which the trained machine-learning model might be used and is pre-specified before starting model training. Various classification-specific evaluation metrics are discussed in Section 2.7; these are used to evaluate the classification techniques described in the later chapters. A typical process of initializing, training, validating, and selecting a classification model depicted in Figure 2.7 is primarily based on Goodfellow et al. (2016) and Zhang et al. (2021) and is described as follows:

1. Preliminary decisions: Before starting the training process, a researcher makes some preliminary decisions. Based on the non-functional requirements for the target application and the available computation hardware, the researcher decides on a model architecture and the first set of hyperparameters. In the case of neural network models, the loss function and optimization algorithm are also decided at this point. The specifics for these are defined in Section 2.6. The evaluation metrics and the stopping criteria for halting the training process are also defined at this stage. In addition, the training, validation, and test splits are defined; these are later used to train the model and evaluate its generalization capability on unseen examples.

2. Pre-processing: Next, the documents in the three data splits are pre-processed and mapped to data objects based on a document model. The pre-processed training, validation, and test splits are represented as (D^{train}, Y^{train}), (D^{val}, Y^{val}), and (D^{test}, Y^{test}), respectively.

3. Model initialization: The document classification model is initialized using the initial set of hyperparameters, which were decided on in Step 1. The model parameters are initialized with random values or with a pre-specified distribution.

4. Model training: Next, the document classification model is trained using the training set (D^{train}, Y^{train}).

5. Model validation: The performance of the trained model is evaluated on the validation set using the evaluation metrics selected in Step 1, and hyperparameter and model checkpoints are saved.

6. A neural network model will be trained multiple times on the same training dataset; a single training cycle is referred to as a training epoch. At the end of a training epoch, the control may proceed in one of three ways:

   (a) Restart training: Based on the model's performance on the validation set, the researcher may initialize a new model with a different set of hyperparameters and restart training.

   (b) Stop training: At this point, the training process might be halted based on predefined stopping criteria. Early stopping is one of the most widely-used stopping criteria: the training process stops if the model prediction on the validation set does not improve for k epochs (Friedrich et al., 2020).

   (c) Continue training: After each epoch, the model is evaluated on the validation dataset, and the training continues if the stopping criterion has not been satisfied.

7. Model selection: Once the training process stops, the researcher can select the best model based on the performance of the classification model on the validation set and evaluate it on the unseen test dataset. The evaluation is performed using the evaluation metrics selected in Step 1.

In this section, we discussed the key concepts associated with the document classification task and then detailed the initialization, training, validation, and model selection processes. Next, we discuss the text representation methods that will later be used in the proposed document representation techniques and baselines.

2.4 Text-Based Representation

As discussed in the previous section, the first step in the document classification pipeline maps a document to a feature vector provided as input to the classification model. The vector can be generated with textual content and associated metadata. Here, we describe some commonly used text representation methods. Based on the improved performance of methods using Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) in various NLP tasks at the Text Retrieval Conference, Craswell et al. (2020) divide text representation methods into two groups: pre-BERT and BERT era methods. In Section 2.4.1, we discuss non-neural text representation techniques, whereas pre-BERT and BERT-based neural text representation methods are described in Sections 2.4.2 and 2.4.3, respectively.

2.4.1 Non-Neural Text Representation

As a simple measure of term importance, *term frequency* counts the number of occurrences of a term within a document. In a corpus with huge variation in document length, term frequency is normalized with the count of terms within a document (Manning et al., 2008, page 117). In Equation 2.1, $f_{t,d}$ is the frequency of a term t in a document d, and the denominator represents the document length.

$$\mathrm{tf}(t, d) := \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{2.1}$$

Term frequency ignores the corpus-level statistics for a term, and therefore, the importance of a term in the corpus is not known. For example, a less frequent term might be more informative than one appearing in most of the documents in the corpus. To account for the corpus-level statistics of a term, Sparck Jones (1988) define a term statistic referred to as inverse document frequency (IDF), which is shown in Equation 2.2. Given the document collection D, the denominator represents the document count for a term t, i.e., the number of documents in which a term t appears.

$$\mathrm{idf}(t, D) := \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{2.2}$$

The product of term frequency and inverse document frequency accounts for the importance of a term within a document and the corpus, and the resulting term vector is referred to as the TF-IDF vector (Equation 2.3).

$$\mathrm{tfidf}(t, d, D) := \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D) \tag{2.3}$$

Due to its ability to represent long text, the TF-IDF approach has been actively used for patent classification (Fall et al., 2003; Benites et al., 2018). Further, in a recent evaluation, Galke and Scherp (2022) found that TF-IDF-based text classification methods are strong baselines.

2.4.2 Neural Text Representation: Pre-BERT

Although efficient, term-count-based feature vectors ignore crucial sequence information and represent the text as a sparse vector of size equal to the term vocabulary. Moreover, they ignore term semantics. Word embedding algorithms capture term semantics and generate representations based on local and global contexts. Here, we describe the key methods for generating word embeddings, then detail widely-used techniques for representing complete sequences: methods based on Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).

Word Embeddings. Deerwester et al. (1990) propose an approach for learning word representations from a document-wide context using latent semantic analysis (LSA). The proposed algorithm learns a latent representation of terms and documents by first representing the corpus as a document-term matrix and then determining term-topic and document-topic vectors by applying matrix factorization. The word2vec model learns word representations based on the local context for a word (Mikolov et al., 2013a). Further, word2vec-based techniques have achieved better results on the sentence completion task (Zweig and Burges, 2011) than state-of-the-art methods, including the LSA-based approach (Deerwester et al., 1990).

The word2vec algorithm is a key development in natural language processing, as it provides a mechanism for efficiently learning word representations with two variants of the auxiliary tasks (Mikolov et al., 2013a). The first variant, the continuous bag of words (CBoW) model, learns word embeddings by predicting the center word using the neighboring words in its left and right contexts as input. In contrast, a second variant, the skip-gram model, learns word representations by predicting the neighboring words with the center word as input. However, word2vec suffers from the out-of-vocabulary issue and can only represent words present in the vocabulary. To some extent, the out-of-vocabulary issue is addressed by the fastText algorithm, which breaks a word into multiple subwords and learns a representation for each subword (Bojanowski et al., 2017). In such cases, the embedding for a word is determined by taking the sum of all the subword embeddings.

Word2vec and fastText provide non-contextualized representations that ignore the current context of a word. For example, the word “apple” has a single representation, independent of its reference to a technology company or a fruit. For this reason, word2vec and fastText embeddings are also referred to as non-contextualized embeddings.

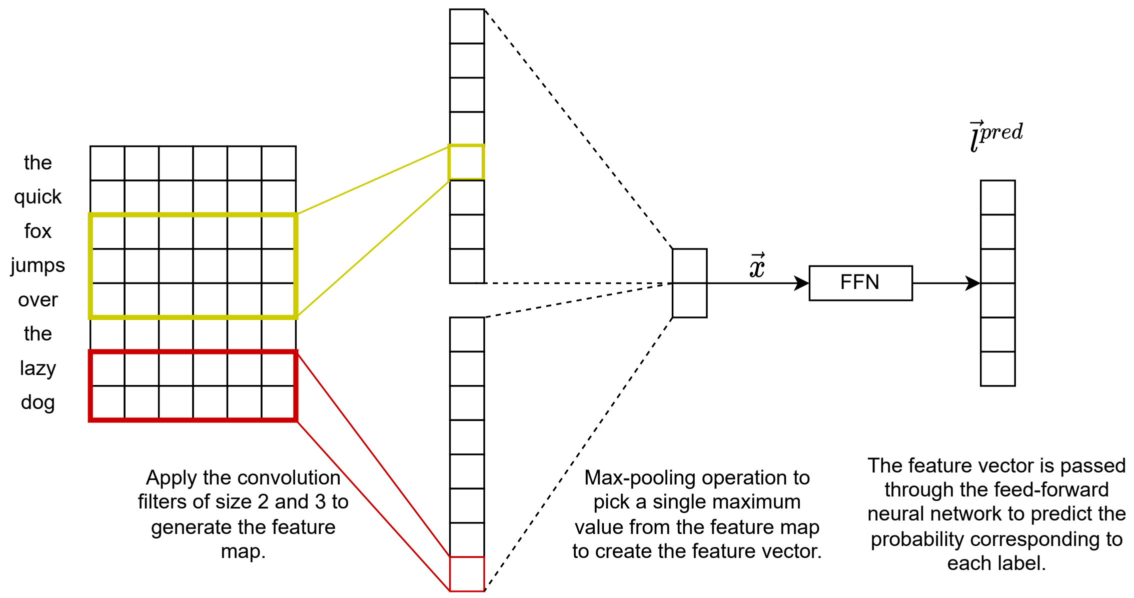

the
quick
fox
jumps
over
the
lazy
dog
Apply the convolution filters of size 2 and 3 to generate the feature map.
Max-pooling operation to pick a single maximum value from the feature map to create the feature vector.
$\vec{x}$
FFN
$\vec{l}^{pred}$
The feature vector is passed through the feed-forward neural network to predict the probability corresponding to each label.

Figure 2.8: Text classification model using a convolutional neural network for feature extraction, based on Kim (2014).

Here, we list a few related works in which word embeddings are used for patent classification and retrieval. Risch and Krestel (2018) train word2vec embeddings using texts from 5M patents and generate a patent classification representation using gated recurrent units (Cho et al., 2014). Hofstätter et al. (2019) propose an approach that combines the document-wide context and the local-window context to learn a word representation for the patent retrieval task.

In the context of text classification, we are interested in aggregating token embeddings in sequence to generate a sequence representation. A simple and less effective method is to take an average of all token embeddings (Iyyer et al., 2015). For text classification, a widely accepted method for representing a sequence is to apply convolutional neural networks (Kim, 2014) or recurrent neural networks (Zhou et al., 2016), or its variants, to a sequence of word embeddings. These two types of neural networks are defined in the following.

Convolutional Neural Networks (CNNs). Convolutional Neural Networks (CNNs) were found to have extensive utility in the field of computer vision (Krizhevsky et al., 2012) before being applied to text classification (Kim, 2014). A typical CNN-based image classification model consists of multiple CNN layers arranged as a cascade. However, the model architecture proposed by Kim (2014) consists of a single CNN layer and is described in the following. A convolution layer consists of several filters. Each filter has a defined *filter width*, which refers to the number of words to which the convolution operation is applied to generate a real-valued output. A convolution filter is applied over the input sequence with a fixed length stride, generating a feature map from which the maximum value is selected with the max-pooling operation. Since the convolution layer consists of multiple filters, multiple feature values are generated, one corresponding to each filter. These are referred to as feature vectors. Figure 2.8 shows an example of a CNN layer with two filters of filter widths 2 and 3, which generates a two-dimensional output feature vector. However, in practice, a CNN layer consists of hundreds of filters. The feature vector is provided as input to the feedforward neural network that predicts the probability distribution over class labels. Targeting the CPC classification task, Li et al. (2018a) propose a model architecture similar to Kim (2014). In Chapter 5, we experiment with a CNN-based architecture consisting of a CNN layer that extracts features from a sequence of token embeddings. Next, we look into recurrent neural networks trained on the next token prediction task.

Recurrent Neural Networks (RNNs) and Their Variants. A language model provides the joint probability of tokens appearing within a text sequence, $\{x_1, x_2, ..., x_T\}$. Based on the Markov assumption, Equation 2.4 provides the joint probability for a sequence where the probability of occurrence of the token x_t is conditioned on the previous $n-1$ tokens; this is also referred to as an n-gram model (Zhang et al., 2021, page 302).

$$P(x_1, ..., x_T) := \prod_{t=1}^{T} P(x_t | x_{t-1}, x_{t-2}, ..., x_{t-(n-1)}) \tag{2.4}$$

There are two main problems with such a probabilistic model. First, the number of parameters increases exponentially with an increase in vocabulary size. Second, no prior information exists for out-of-vocabulary terms.

Recurrent neural networks (RNNs) address these issues with a model architecture that incorporates a hidden state variable to keep track of previous tokens, as shown in Equation 2.5 (Elman, 1990). The hidden state $\boldsymbol{h}_t$ is a function of the input vector $\boldsymbol{x}_t$ and the hidden state $\boldsymbol{h}_{t-1}$, parameterized with the matrices $\boldsymbol{W}_h$ and $\boldsymbol{U}_h$ and the vector $\boldsymbol{b}_h$. Here, $\sigma_h(.)$ is the non-linear activation function, e.g., tanh. The output vector $\boldsymbol{o}_t$ is computed by multiplying the parameter matrix $\boldsymbol{W}_o$ to the hidden state $\boldsymbol{h}_t$.

$$\begin{aligned} \boldsymbol{h}_t &:= \sigma_h(\boldsymbol{W}_h \boldsymbol{x}_t + \boldsymbol{U}_h \boldsymbol{h}_{t-1} + \boldsymbol{b}_h) \\ \boldsymbol{o}_t &:= \boldsymbol{W}_o \boldsymbol{h}_t + \boldsymbol{b}_o \end{aligned} \tag{2.5}$$

RNNs suffer from the vanishing gradient problem, which means that the parameter updates for a longer sequence become insignificant as the gradient value becomes too low. To address this issue, Hochreiter and Schmidhuber (1997) propose a long short-term memory (LSTM) architecture that consists of a memory cell to hold a value for an arbitrary time interval. The flow of information to and from the memory cell is controlled using three gates: input, output, and forget. Cho et al. (2014) propose a similar but simplified solution, which they call gated recurrent units (GRUs). For certain use cases, e.g., text classification, it can be assumed that both the left and right contexts for the tokens are known. Therefore, a representation can be generated using both contexts with bi-directional RNNs (Schuster and

Paliwal, 1997). RNNs variants are often used for text representation in the context of patent classification tasks, for which LSTM is a popular choice (Grawe et al., 2017; Shalaby et al., 2018; Hu et al., 2018) and GRUs is sparingly used (Risch and Krestel, 2019).

2.4.3 Neural Text Representation: BERT

Recurrent models are inherently sequential as the computation of the current state h_t is a function of the previous step h_{t-1}, an operation that cannot be parallelized. It is also challenging to capture the long-range dependencies between input tokens. Vaswani et al. (2017) address these issues with transformer architecture that uses the attention mechanism instead of recurrence and computes the contextualized representation based on the attention weights corresponding to all other tokens in the sequence; this is also referred to as the full-attention mechanism. Since the recurrence operation is not involved, the key computation steps are expressed as a set of matrix operations, which can be parallelized.

A transformer consists of multiple attention units or *attention heads*, each of which computes the representation of input tokens using the *scaled-dot product* operation. This is performed using three matrices W^Q, W^K, and W^V, referred to as trainable *query*, *key*, and *value* matrices. An input embedding corresponding to the i^{th} token is multiplied with W^Q, W^K, and W^V to generate query ($\boldsymbol{q}_i$), key ($\boldsymbol{k}_i$) and value ($\boldsymbol{v}_i$) vectors, respectively. To compute the attention weight a_{ij}, first, a dot product is computed between the query vector q_i and the key vector, k_j (corresponding to the i-th and j-th tokens, respectively), which is then normalized with the dimension of the key vectors $\sqrt{d_k}$ and finally passed through a softmax function. Representing the query, key, and value vectors as matrices Q, K, and V, in which the i-th row corresponds to the i-th token, the attention function can be defined as follows:

$$\text{Attention}(Q, K, V) := \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V \tag{2.6}$$

As depicted in Equation 2.6, the representation of a token is computed by taking the attention-weighted sum of all the other tokens in the sequence. The three trainable matrices in each attention head capture different relationships between input tokens. The final representation is generated by concatenating the outputs from all the attention heads, as shown in Equation 2.7. Here, W^O is the parameter matrix, and [...] represents the vector concatenation operation.

$$\begin{aligned} \text{MultiHeadAttention}(Q, K, V) &:= \text{Concat}[\text{head}_1; \cdots ; \text{head}_n]W^O \\ \text{with: } \text{head}_i &:= \text{Attention}(QW_i^Q, KW_i^K, VW_I^V) \end{aligned} \tag{2.7}$$

Devlin et al. (2019) take the encoder part of the encoder-decoder architecture proposed by Vaswani et al. (2017) and propose a language model that is trained on masked-token

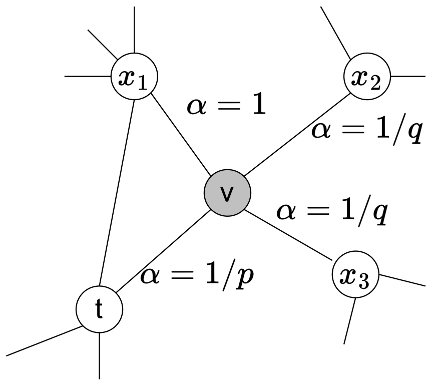


Figure 2.9: The next node selection step for node2vec, based on Grover and Leskovec (2016), depends on the search bias $\alpha_{pq}(t, v)$ parameter which determines the weight for an edge based on the shortest distance between the previous node t and neighbors of the source node v (Equation 2.8).

prediction and next-sentence prediction tasks. The proposed model architecture is called Bidirectional Encoder Representation from Transformers (BERT) and is pre-trained with BookCorpus (Zhu et al., 2015) and English Wikipedia text. The vocabulary corresponding to a pre-trained model depends on the training dataset used, and therefore, BERT is non-ideal for patent-related tasks. For this reason, we use SciBERT (Beltagy et al., 2019), a BERT variant that is pre-trained on PubMed articles and computer science publications and thus has a vocabulary more similar to patents as compared to BERT.

In the context of a classification task, a representation can be generated for the input text by performing the following steps. As a first step, the input text is tokenized. The token sequence is truncated if its length is more than the maximum allowed sequence length. Two special tokens, [CLS] and [SEQ], are added at the beginning and end of the sequence, respectively, while keeping the sequence length less than or equal to the maximum allowed sequence length value. Next, the sequence is provided as input to the pre-trained language model, which maps tokes to embeddings. The embeddings are passed through multiple stacked transformer layers. The output of the final hidden layer for the [CLS] token is usually used as a sequence representation (Lee and Hsiang, 2019). In this thesis, the method $e_f(.)$ denotes the sequence embedding generation process, where the subscript f refers to the case when the pre-trained language model parameters are fine-tuned.

2.5 Graph-Based Representation

A document consists of content and metadata semantic elements arranged at different levels of granularity, with the document being the highest-level semantic element (Ingwersen, 1994). The content and metadata elements can be represented as a corpus-level graph in which the nodes correspond to the semantic elements, and the edges represent the implicit or explicit relations between them. Representations for the semantic elements can then be learned with graph embedding techniques.

For example, TextGCN (Yao et al., 2019) represents words and documents as graph nodes linked by a co-occurrence relation, from which it can learn an embedding for documents and words. However, when evaluated on text classification tasks, Galke and Scherp (2022) found that graph-based text representation techniques are less effective than pre-trained transformer models.

In addition to the content semantic elements, metadata information across documents can also be represented using graphs. For example, the CPC labels associated with patents in a patent corpus can be represented as a graph in which the nodes represent the CPC labels and a pair of nodes are connected with an undirected link representing the co-occurrence relation, i.e., the relation constituted by the two CPC labels occurring in the same patent. A graph embedding method can then be applied to learn a label representation.

Chen et al. (2020) provide an extensive survey of graph embedding methods and list DeepWalk (Perozzi et al., 2014) and node2vec (Grover and Leskovec, 2016) as two key methods that learn node embeddings by first performing multiple random-walks to generate node sequences which are then provided as input to the word2vec algorithm. However, the two algorithms differ in the random walk strategies they apply. DeepWalk favors a depth-first sampling (DFS) strategy, traversing nodes further away from the source node, whereas node2vec traverses nodes with both DFS and breadth-first sampling (BFS) strategies.

Node2vec generates a sequence of nodes with multiple random walks (Grover and Leskovec, 2016). The hyperparameter *number of walks* defines the number of random walks originating from each node in the graph. In contrast, the *walk length* parameter controls the number of nodes traversed in a random walk. Node2vec defines hyperparameters, p and q, that control the traversal strategy (BFS or DFS) for selecting the next hop.

Given t and v as the previous and source (current) nodes, respectively, the next hop from the neighboring nodes is selected with Equation 2.8, where d_{tx} is the shortest path distance between the previous node t and the neighbors of v. As seen in Figure 2.9, the edge weights are calculated with Equation 2.8, which represents the search bias $\alpha_{pq}(t, v)$ when identifying the next hop. The unnormalized transition probability is given as $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$, where w_{vx} is the weight for the edge (v, x).

$$\alpha_{pq}(t, v) := \begin{cases} \frac{1}{p}, & \text{if } d_{tx} = 0 \\ 1, & \text{if } d_{tx} = 1 \\ \frac{1}{q}, & \text{if } d_{tx} = 2 \end{cases} \tag{2.8}$$

As per Equation 2.8, a lower value of p favors the previously traversed node as the next hop, whereas a lower value of q tends to result in the selection of nodes further away from the source, thus favoring the DFS strategy. In Chapter 7, we generate CPC/IPC label embeddings by applying the node2vec algorithm over the undirected label co-occurrence graph and incorporating them into patent representation (see Section 7.3). The label co-occurrence graph consists of CPC/IPC labels as nodes, and two labels are connected if they both occur in one patent.

2.6 Classification Model

A classification model takes a feature vector as input and maps it to a label vector, representing the probability distribution corresponding to target labels. In this thesis, we are interested in parametric models, which consist of a set of parameter variables that determine the labels for an input feature vector and are learned during the model training process. Here, we discuss the details of non-neural and neural classification models often used in patent classification tasks.

2.6.1 Non-Neural Classifiers

Here, we provide a brief overview of two key non-neural classification models: Logistic Regression and Support Vector Machines. Logistic regression is a simple and interpretable classification model that separates the positive and negative classes with a linear decision boundary. Logistic regression has long been used as a tool for statistical analysis (Truett et al., 1967; Priyadarshini et al., 2021) and mentions of the model can be found in early textbooks on the analysis of medical data (Pearl, 1924). However, because the logistic regression's decision function is restricted to a space of linear functions, it has been followed by the development of more sophisticated classification models, such as Support Vector Machines (Cortes and Vapnik, 1995). Below, we discuss the key concepts of both of these models.

Logistic Regression. Logistic regression is a classification model that represents the decision boundary between the positive and negative classes as a linear function (Ng, 2022, page 8). For an input vector $\boldsymbol{x}$, a value is predicted by first taking a dot product between $\boldsymbol{x}$ with the parameter weight vector $\boldsymbol{\theta}$ and then mapping the dot product to a value between 0 and 1 with the sigmoid function. Given that the values across feature vectors are within the same range, the weight value θ_j represents the importance of feature x_j when performing the classification. The decision function is as follows:

$$h_\theta(\boldsymbol{x}) := \frac{1}{1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}}} \tag{2.9}$$

The model weights are learned with the training set. The weight update step for the j^{th} parameter is represented in Equation 2.10, where $(y^{(i)} - h_\theta(\boldsymbol{x}^{(i)}))x_j^{(i)}$ is the error term, and α is the learning rate, a hyperparameter that controls the weight updates (Ng, 2022, page 12).

$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(\boldsymbol{x}^{(i)}))x_j^{(i)} \tag{2.10}$$

Equation 2.10 is for the stochastic gradient descent algorithm that randomly picks one example, computes the gradient, and updates the model weights (Ng, 2022, page 13). In practice, the mini-batch gradient descent algorithm, which computes the gradient for m examples at a time, is commonly used; it results in smoother gradient updates and is more efficient.

Support Vector Machines. Logistic regression has several key issues. First, due to the possibility of multiple decision boundaries existing between the data points from two classes, there is no precise mechanism for picking the optimal one. Second, logistic regression can only estimate a linear decision boundary and performs poorly in cases of non-linear decision boundaries. Support Vector Machines (SVM) address these issues using the following mechanisms (Cortes and Vapnik, 1995).

First, instead of a separating hyperplane, an SVM has a separation region bounded by margins on either side. The hyperplane is at the center of the separation region, equidistant from the two margins. The separation region is estimated by solving the optimization problem that maximizes the distance between the hyperplane and the margins and minimizes the classification error, thus enabling the SVM to estimate the optimal hyperplane (Cortes and Vapnik, 1995). The optimization function defines a parameter controlling the number of support vectors (Cortes and Vapnik, 1995). The decision function predicts a value in the range $[-1, +1]$ and is represented by Equation 2.11.

$$f(\boldsymbol{x}) := Sgn(\sum_{i} \alpha_i y_i(\boldsymbol{x} \cdot \boldsymbol{x_i}) + b) \tag{2.11}$$

Here, $\boldsymbol{x}_i$ is the feature vector, y_i is the target value, and α_i is the value of the trainable parameter for the i^{th} training instance. The value of α_i is determined during the optimization step. The method $Sgn(.)$ determines the sign of the output value. A value of $f(\boldsymbol{x}) > 0$ predicts a positive class, else the negative class is predicted. The data points in the training dataset that are on the margin of the decision boundary are important as they mark the boundary of the separation region and are referred to as support vectors.

Further, unlike logistic regression, SVMs can learn a non-linear decision boundary. The feature vector is mapped to a high-dimensional space to learn the non-linear decision boundary, assuming that transformed vectors might be linearly separable (Cortes and Vapnik, 1995). The decision and objective functions involve a dot product between two very high-dimensional feature vectors, which is computationally infeasible (for more details, see Cortes and Vapnik (1995)). To address this issue, an SVM uses the kernel function $K(.)$ to represent the dot product as a function of the corresponding vectors in the original space, such that $\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}_i) = K(\boldsymbol{x}, \boldsymbol{x_i})$. With this change, the decision function takes the following form:

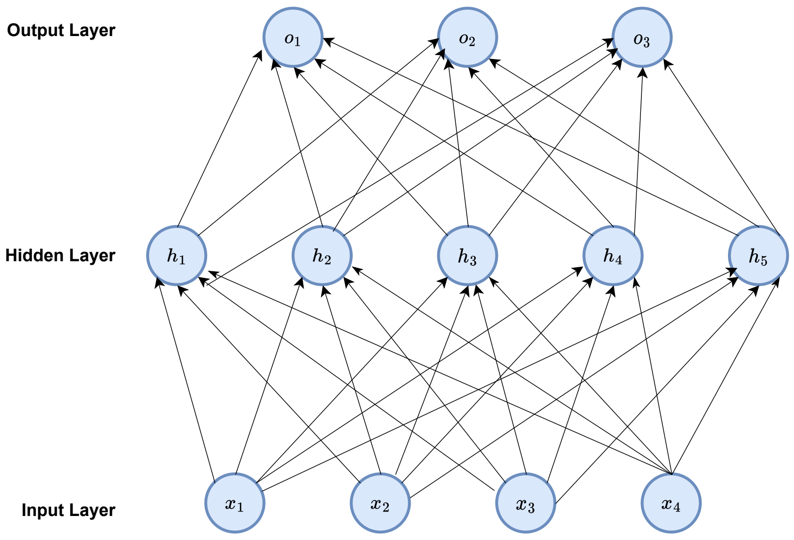


Figure 2.10: A multilayer perceptron architecture based on Zhang et al. (2021).

$$f(\boldsymbol{x}) := Sgn(\sum_i \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x_i}) + b) \tag{2.12}$$

The shape of the non-linear decision boundary depends on the kernel method and is therefore decided a priori. A feedforward neural network can estimate an arbitrary shape decision boundary, as discussed in the next section.

2.6.2 Multilayer Perceptron

Multilayer Perceptron (MLP) models generalize the idea of the perceptron (Rosenblatt, 1958) to a multilayer, fully connected neural network architecture (Zhang et al., 2021, Ch. 5). Figure 2.10 shows an example of an MLP architecture: a two-layer network. The input layer does not perform any computation and has a dimension equal to that of the feature vector $\boldsymbol{x}$. The other two layers consist of multiple computation units, each with its own parameter vector and activation function. A computation unit gets a vector from the previous layer as input, computes its dot product with the parameter vector, and applies the activation function to generate a numeric output. The output of a neural layer is a vector that gets a value from each computation unit. The details of the operation of the hidden and output layers are as follows.

Equation 2.13 represents the computation step for the hidden layer of MLP in Figure 2.10. In the context of a document classification task, $\boldsymbol{x}$ is the document representation, $\boldsymbol{W}^{(1)}$ and $\boldsymbol{b}^{(1)}$ are the weights and biases for the hidden layer, $\sigma(.)$ is the activation function, and $\boldsymbol{h}$ is the output of the hidden layer. $ReLU$ is commonly used as an activation function for hidden layers, as it mitigates the vanishing gradient problem (Zhang et al., 2021, page 185).

$$\boldsymbol{h} := \sigma(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}) \tag{2.13}$$

The output from the hidden layer is provided as input to the output layer. Equation 2.14 shows this computation step. Here, $\boldsymbol{W}^{(2)}$ and $\boldsymbol{b}^{(2)}$ are the output-layer weights and biases, and $sigmoid(.)$ is the sigmoid activation function applied to each computation unit in the output layer. In the context of patent classification tasks, the i^{th} value of the output vector $\boldsymbol{o}_i$ corresponds to the prediction probability of label $l_i \in L$.

$$\boldsymbol{o} := sigmoid(\boldsymbol{W}^{(2)}\boldsymbol{h} + \boldsymbol{b}^{(2)}) \tag{2.14}$$

As with logistic regression, the optimal parameter values are estimated by iteratively applying an optimization algorithm that computes the gradient of loss and adjusts the model weights. However, the output of a neural network is computed by applying multiple vector operations and functions, and therefore, the gradient computation is not straightforward. The neural network parameters are updated with the backpropagation mechanism, in which the computation graph is traversed in the reverse order, from output to input. At each step, gradients are computed by employing the chain rule from calculus (for details, see (Zhang et al., 2021, page 181)).

2.6.3 Hierarchical Classification Models

The classification models discussed above are categorized as *flat classification* approaches, in which the labels from one level of the hierarchy, often the leaf labels, are predicted. However, the CPC/IPC classification task is inherently hierarchical as it involves predicting labels at different levels. In this case, a hierarchical classifier that predicts labels across the hierarchy might perform better by learning to predict labels representing concepts at varied levels of granularity. Silla and Freitas (2011) categorized hierarchical classification approaches into two main categories: the *local classifier per level* approach (LCL) and the *local classifier per node* approach (LCN). An LCL approach trains a classifier for each level in the class hierarchy. Given the example taxonomy illustrated in Figure 2.11, an LCL approach trains three classifiers, one for each level. In contrast, an LCN approach trains a classifier for each label in the taxonomy.

TwistBytes. TwistBytes is a non-neural LCN-based approach that trains a classifier for each label in the taxonomy (Benites, 2019). Further, it employs the sibling strategy to train a label-specific classifier that filters training instances that have the label or its siblings as the target. For example, given the taxonomy shown in Figure 2.11, when training a classifier for label “A43B” only instances containing either “A43B” or its siblings, “A43C”, as target labels are considered.

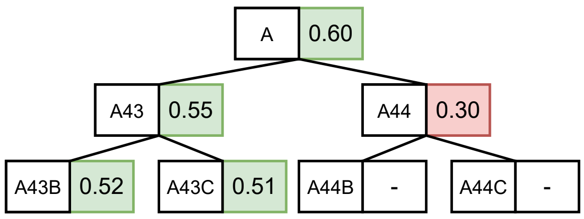


Figure 2.11: The example shows CPC labels up to the third level of the hierarchy. Each label has an associated prediction score based on the prediction of the corresponding label-specific classifier. A label is predicted if the prediction for the parent label exceeds a pre-defined threshold. In this case, "A44B" and "A44C" are not predicted as the prediction score for the parent "A44" is less than the pre-defined threshold of 0.50.

The tree is traversed from top to bottom when predicting labels for an input representation. A label-specific classifier predicts whether the label applies to an input representation at each hop. Further, for level two and below, a label is predicted if the prediction score for the parent label exceeds a pre-defined threshold. When predicting labels for the taxonomy shown in Figure 2.11 with a prediction threshold value of 0.50, TwistBytes does not predict labels "A44B" and "A44C" because the prediction score for the parent, "A44", is less than the pre-defined threshold.

Hierarchical Multi-label Classification Network (HMCN). Wehrmann et al. (2018) propose a neural LCL-based model architecture that trains a classification head for each level in the class hierarchy and a global classification head that predicts probability distribution for all the labels in the taxonomy. The final prediction probability is a weighted sum of predictions from the level-specific and global heads. For the sample taxonomy shown in Figure 2.11, an HMCN model would train four classification heads: one level-specific classification head for each level and a global classification head. The model architecture for the taxonomy of Figure 2.11 is shown in Figure 2.12.

A level-specific classification head consists of two dense layers. The first dense layer has a ReLU activation, whereas the second dense layer has a sigmoid output predicting probabilities corresponding to all labels in the level. The classification head corresponding to the first level takes the feature vector $\boldsymbol{x}$ as input. For each subsequent level, the hidden state of the first dense layer from the classification head of the previous layer is concatenated with $\boldsymbol{x}$ and provided as input. The input for the global classification head is generated by concatenating the hidden state of the classification head corresponding to the lower-most level to the feature vector $\boldsymbol{x}$.

$$\boldsymbol{l}^{pred} := ((1-\beta) \times (\boldsymbol{l}_1^{pred}; \boldsymbol{l}_2^{pred}; \boldsymbol{l}_3^{pred})) \oplus (\beta \times \boldsymbol{l}_G^{pred}) \tag{2.15}$$

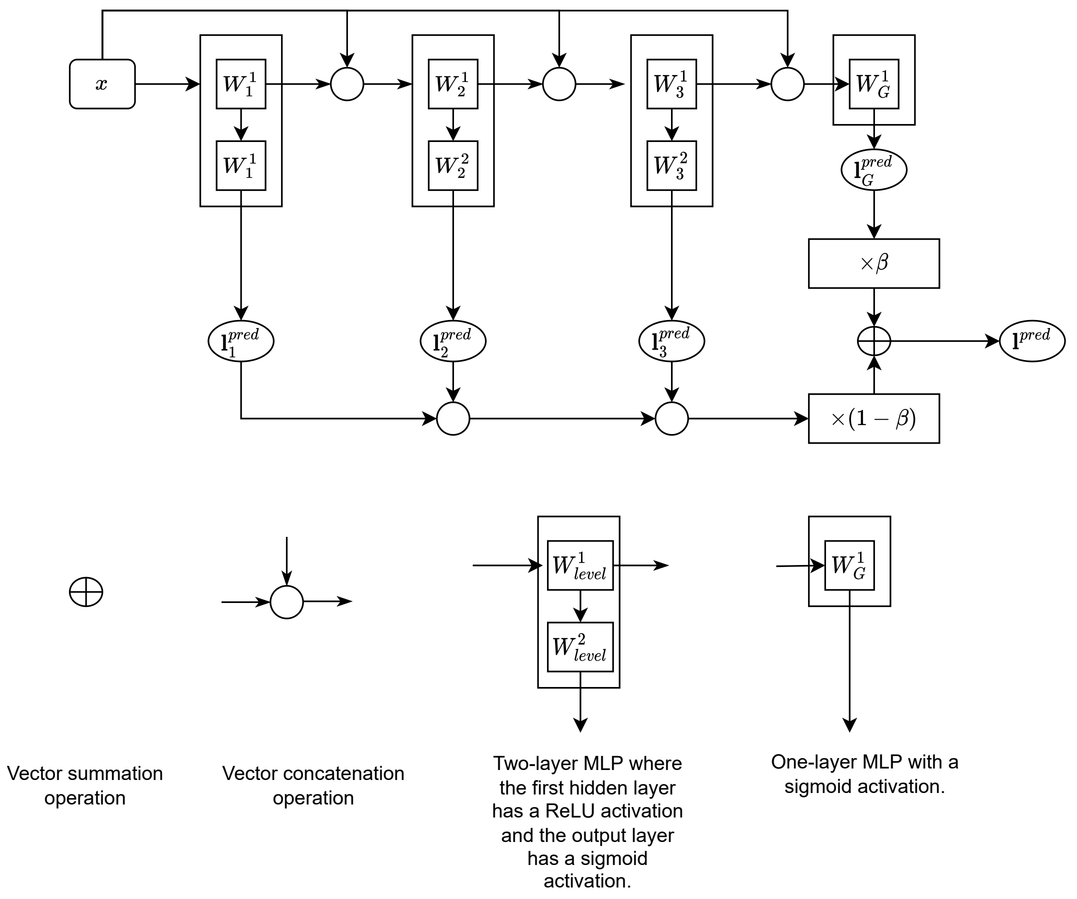

x
W_1^1
W_1^1
W_2^1
W_2^2
W_3^1
W_3^2
W_G^1
l_G^{pred}
$\times\beta$
l_1^{pred}
l_2^{pred}
l_3^{pred}
l^{pred}
$\times(1-\beta)$
W_{level}^1
W_{level}^2
W_G^1
Vector summation operation
Vector concatenation operation
Two-layer MLP where the first hidden layer has a ReLU activation and the output layer has a sigmoid activation.
One-layer MLP with a sigmoid activation.

Figure 2.12: Hierarchical Multi-label Classification Network (HMCN) architecture, based on Wehrmann et al. (2018).

The final prediction $\boldsymbol{l}^{pred}$ is a weighted sum of level-specific predictions, $\boldsymbol{l}_1^{pred}$, $\boldsymbol{l}_2^{pred}$, and $\boldsymbol{l}_3^{pred}$, and predictions from the global classification head, $\boldsymbol{l}_G^{pred}$, weighted with parameter β as shown in Equation 2.15. The semicolon (;) and $\oplus$ represent the vector concatenation and summation operations, respectively.

Hierarchical Attention-based Recurrent Neural Network (HARNN). Huang et al. (2019) propose a hierarchical classifier similar to HMCN, which consists of multiple hierarchical attention-based memory (HAM) units, one for each level. As a first step, the document text is represented with a BiLSTM layer that provides an embedding corresponding to each token, whereas the labels in the taxonomy are represented as level-wise matrices, one for each level. HAM takes BiLSTM-based token representations, a level-wise label matrix, and representation from HAM in the previous level as input and computes a level-wise prediction P_L^h based on attention score between the text representation and level-wise label matrix, where h is a hierarchical level. The output of the HAM corresponding to the lower-most level is passed through a feedforward network to compute global prediction P_G for all the labels in the hierarchy. The final prediction for a label is the weighted sum of the corresponding prediction scores in P_L^h and P_G.

2.7 Evaluation Metrics

We now discuss the evaluation metrics that capture different aspects of a classifier's performance, such as precision, and can be useful for determining its usability for an application. Given a set of true labels L_d and predicted labels $\hat{L}_d$, we are interested in true positive, false negative, and false positive sets computed with set operations $L_d \cap \hat{L}_d$, $L_d \setminus \hat{L}_d$, and $\hat{L}_d \setminus L_d$, respectively. The numbers of instances in true positive, false negative, and false positive sets are referred to as TP, FN, and FP, respectively.

Precision, Recall, and F1-score. Precision, recall, and F1-score are commonly used evaluation metrics for evaluating classification tasks and are described in the following based on Davis and Goadrich (2006). In the context of a classification task, a precision score evaluates the ability of the classification model to make a precise judgment, i.e., by predicting fewer false positive labels (Dogan et al., 2019). Instead of making a precise judgment, in certain cases, we aim to maximize the prediction of relevant labels, i.e., to maximize true positives even if doing so increases false positives. For example, when performing a patent landscape study to identify all patents relevant to an invention, a domain expert intends to minimize false negatives, as missing relevant patents from competitors might lead to litigation. Recall is an important metric in such situations. It is defined as the ratio of correctly predicted labels to the actual label count (Equation 2.16).

$$\begin{aligned} Precision &:= \frac{TP}{TP + FP} \\ Recall &:= \frac{TP}{TP + FN} \end{aligned} \tag{2.16}$$

Considering these definitions of precision and recall scores, we find that one can be improved by impacting the other. In a multi-label patent classification setting, recall could be maximized by predicting more labels than the average label count value. However, such a prediction strategy would lead to poor precision. Similarly, precision could be maximized by limiting the number of predicted labels. An appropriate balance between the precision and recall is desirable and is provided by the F_β-score (F_β) defined in Equation 2.17. The β parameter controls the influence of precision and recall. The precision and recall scores are balanced by giving equal weights to precision and recall with a β value of 1, referred to as the F1-score (Equation 2.17).

$$\begin{aligned} F_\beta\text{-score} &:= (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot (Precision + Recall)} \\ F_1\text{-score} &:= 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \end{aligned} \tag{2.17}$$

Macro-F1. When referring to the F1-score, we often refer to the micro-F1, which is the average of F1-scores over all the instances in the dataset. In the case of an unbalanced dataset, the micro-F1 score can be improved by predicting labels corresponding to the majority classes. Often, for a large taxonomy such as the CPC taxonomy (Section 2.1.3), the labels corresponding to less frequent classes are more informative than those corresponding to more frequent classes, which may reflect the appearance of generic concepts in most documents. Opitz and Burst (2019) propose the macro-F1 score metric, which computes the average of F1-scores over all the classes (Equation 2.18).

$$\text{macro-F1} := \frac{1}{|L|} \sum_{class \in L} F_{1_{class}} \tag{2.18}$$

Hierarchical Evaluation Metrics. The pre-defined patent taxonomies are hierarchical in structure (Section 2.1.3). A hierarchical classifier may predict correct labels at a higher level, but we cannot be confident that it will predict lower-level, fine-grained labels. Further, a flat classifier (a classifier predicting labels at a particular hierarchy level) can make a partially correct prediction. The hierarchical evaluation metric captures a classifier's ability to make a partially correct prediction. Silla and Freitas (2011) propose hierarchical evaluation metrics to determine the performance of classification models when predicting labels across levels. In this case, L_d contains all true labels, including ancestors, and $\hat{L}_d$ consists of all predicted labels and their respective ancestors. For example, with flat evaluation metrics, a classifier predicting $\{A43B\}$ for $\{A43C\}$ as a true label results in a precision score of 0, whereas with a hierarchical evaluation metric predicting $\{A, A43, A43B\}$ for the true value set $\{A, A43, A43C\}$ results in a precision score of 0.66.

Chapter 3

Conceptual Model for Document Classification

The document classification techniques developed in this thesis are primarily evaluated for the patent classification task. However, we intend to ensure their applicability for other domains and document types. To achieve this, we propose a conceptual model that represents key components and their relationships. A conceptual model provides a unified view of the classification system, allowing for easy comparison and adaptation for tasks and use cases other than patent classification.

Maass and Storey (2021) emphasized that conceptual modeling is crucial when designing and developing systems involving a machine learning component, citing the following reasons. Firstly, by providing a unified view, conceptual models enable discussions between professionals, such as business analysts, data scientists, data engineers, and developers. Secondly, conceptual models help stakeholders to make crucial design decisions by representing key elements, including quality attributes. Thirdly, conceptual models can represent key machine learning algorithms and their influence on performance indicators and quality attributes.

To conceptualize business solutions involving machine learning components, Nalchigar and Yu (2020) propose the Goal-oriented Requirements Engineering for Machine Learning (GR4ML) framework.[1] Using the GR4ML framework, we describe a conceptual model for document classification in Section 3.1. We further expand a few of the components of the conceptual model to define a classification pipeline that can, with minor modifications, be adapted to perform multiple classification tasks (Section 3.2).

[1]http://www.cs.toronto.edu/~soroosh/gr4ml_introduction.html [last accessed December 10, 2023]

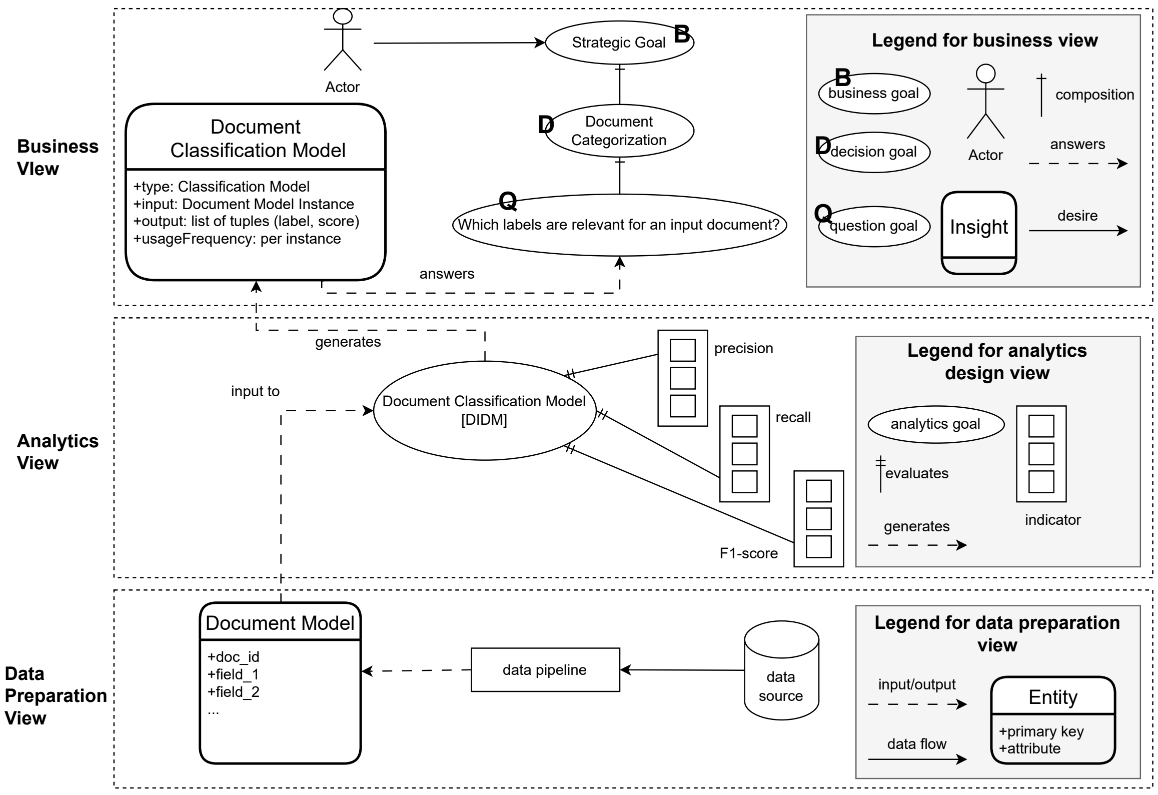


Figure 3.1: Conceptual model based on GR4ML (Nalchigar et al., 2021) for a business analytics use case involving a document classification task.

3.1 Conceptual Model

The GR4ML conceptual modeling framework (Nalchigar and Yu, 2020) can be used to represent three perspectives on the document classification process: business, analytics, and data preparation. Figure 3.1 depicts a conceptual model for document classification, which is comprised of these three views, as described in the following.

Business View. An organization invests in a business analytics solution to achieve its strategic objectives. Within the GR4ML framework, the strategic objectives are represented as business goals in the business view, which are then mapped to one or more decision goals. These decision goals enable an organization to make certain decisions that can help them achieve corresponding business goals. The decision goals are further mapped to a set of question goals, which are answered using the insight element instance generated by the analytics view.

Business goals represent strategic business objectives and are use-case specific. For instance, when automating the CPC classification process, one of the central business objectives for a patent examination office is to reduce the manual effort expended to assign new applications to the appropriate departments (Krier and Zaccà, 2002). In contrast, patent landscape studies are primarily motivated to draw crucial insights by analyzing patents within a field, for which patent categorization is an important pre-step (Abood and Feltenberger, 2018).

In both cases, a domain expert has to decide to assign a set of labels to patents as one of the tasks. This decision-making step is represented as the decision goal in Figure 3.1. This decision goal is then mapped to a question goal, which is answered with the insight generated from the analytics view. For the examples discussed above, it is essential to know which labels are most relevant to an input patent.

Analytics View The analytics view encompasses the analytical components of a business solution, such as algorithms, models, and indicators. The algorithms and models are applied to a data instance based on a defined document model to generate an insight for the business view. In the context of document classification, the document classification model is a crucial component of the analytics view. This takes a document model object as input and generates label scores as insight. Since multiple document classification models may be applicable to a given task setting, the optimal model is selected according to its performance in terms of key performance indicators and quality attributes. Figure 3.1 depicts precision, recall, and F1-score as indicators for evaluation, which are discussed in Section 2.7.

Data Preparation View. The data preparation view comprises three main components: a document model, a data processing pipeline, and a data source. The document model represents the structure of an input document and may vary across document types. A data object based on a document model is provided as input to the document classification model in the analytics view. The data processing pipeline is an abstraction of the data processing step that generates an instance of a document model from the data source, e.g., a database, file system, or web API.

3.2 Classification Pipeline

The document classification pipeline described in the previous section abstracts several crucial components. We extend the conceptual model by expanding some of the components and defining crucial data models to enable its applicability for multiple domains using diverse representation methods.

3.2.1 Pipeline Structure

The overall pipeline is depicted in Figure 3.2 and is divided into three parts. The first part of the pipeline consists of use-case-specific components. As a first step, the dataset is loaded from the data source to create an object of Domain-Specific Document Model (DSDM) that contains information specific to a document type. In Section 3.2.2, we define the specifications for the Patent Document Model (PDM), which represents the structure

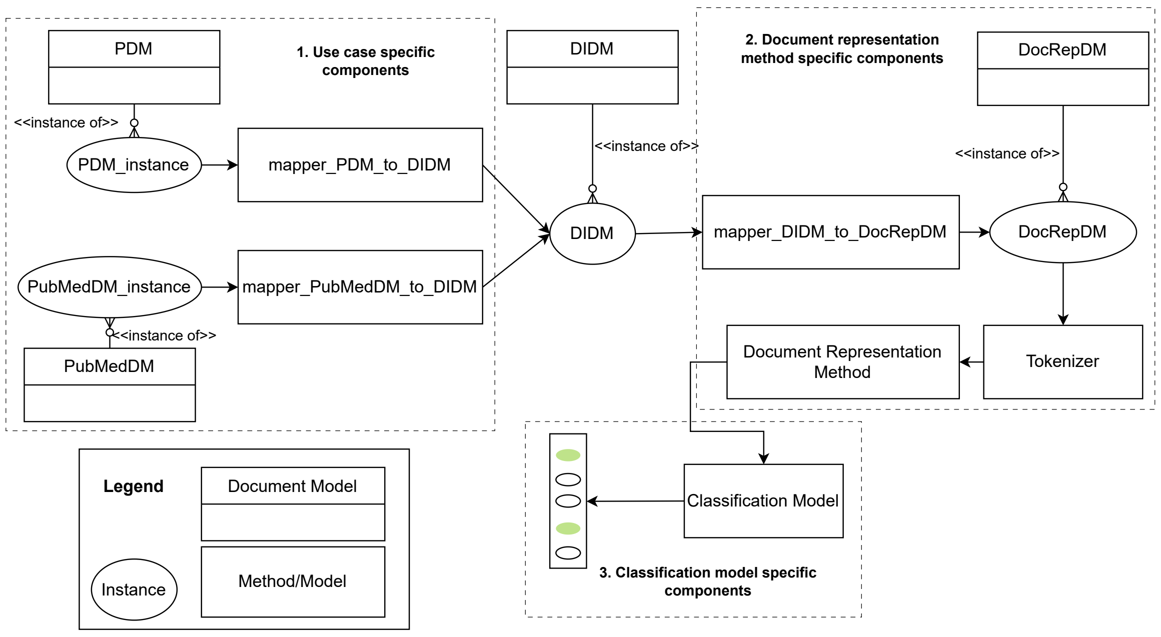


Figure 3.2: Extending the document classification pipeline so that it can be applied for multiple document classification tasks with minor adaptations (Section 3.2.1). The patent document model (PDM) and PubMed document model (PubMedDM) are defined in Section 3.2.2, the domain-independent document model (DIDM) is defined in Section 3.2.3, and the data models for document representation methods (DocRepDM) are described in Section 3.2.4.

and content of a typical patent. Further, to emphasize the applicability of the pipeline for other domains and document types, we adapt the pipeline for the classification of PubMed articles on different categories of COVID-19 topics in Chapter 7. Thus, in addition to PDM, we define a document model for PubMed articles, which is henceforth referred to as PubMedDM (Section 3.2.2).

The DSDM instance is mapped to a Domain-Independent Document Model (DIDM) instance using a mapper function specific to the DSDM type. For the examples shown in Figure 3.2, mapper_PDM_to_DIDM and mapper_PubMedDM_to_DIDM are mappers for PDM and PubMedDM instances, respectively. DIDM is intended to represent a document as a generic data model, such that once a DIDM instance is created, the remaining operations can be applied regardless of the domain and document type. In Section 3.2.3, we define the DIDM by considering the common structural patterns that occur across DSDMs.

As depicted in Figure 3.2, the second part of the pipeline consists of components specific to a document representation method. First, mapper_DIDM_to_DocRepDM maps a DIDM instance to an object based on DocRepDM, a data model specific to the document representation method. With a DocRepDM object as input, the tokenizer generates an output that can be provided as input to the document representation method.

For example, with BERT as the document representation method, we map a DIDM instance to a DocRepDM instance, which consists of text and max_length as attributes.

Next, with the $\mathrm{DocRepDM}$ instance as input, the sequence tokenizer generates a tokenized sequence of the length specified by the $\mathrm{max\_length}$ attribute. Finally, the tokenized sequence is provided as input to BERT.

When adding a new document representation method to the catalog, the corresponding $\mathrm{mapper\_DIDM\_to\_DocRepDM}$, $\mathrm{DocRepDM}$, and $\mathrm{tokenizer}$ need to be updated as well. In Section 3.2.4, we discuss a few types of $\mathrm{DocRepDM}$ which are relevant to the document representation techniques discussed in the subsequent chapters. We define a few crucial attributes for $\mathrm{DocRepDM}$ that can be further extended based on the requirements.

The third part of the pipeline consists of a classification model as the only component that maps the document representation output to a vector of prediction scores corresponding to the target labels. The output of a document representation method should adhere to the interface definition of a classification model. For example, in a typical BERT-based classifier, the text sequence is represented as a single vector, which is provided as input to the feedforward neural network (Devlin et al., 2019). In contrast, a label-based classification technique generates multiple vector representations for the input text, which are provided as input to the label-specific classification heads (Mullenbach et al., 2018).

3.2.2 Domain-Specific Document Models

A previous paper by Kimura and Shaw (1984) introduced a document model independent of the storage medium and presentation view. The World Wide Web Consortium (W3C) proposed a Document Object Model (DOM)[2] to represent Hypertext Markup Language (HTML) and Extensible Markup Language (XML) documents. The W3C's DOM represents the hierarchical structure of these document types with the document object at its root. Further, the Unified Modeling Language (UML)[3] is often used to depict the structure of a document in terms of entities and their relationships (Morbach et al., 2008). In contrast, to address event and entity search as an application, Spitz et al. (2020) represent a document corpus as a hypergraph in which the nodes correspond to different element types (e.g., document, passage, sentence, or tokens) and are connected with a co-occurrence relation. Giereth (2013) proposes a semantic document model targeting visual patent analysis as an application. We want to represent hierarchical document structures in the context of patent classification such that the information can be aggregated bottom-up to generate a document representation and find a document model similar to DOM appropriate for this. Next, we describe the patent document model before discussing the specifications of the PubMed document model.

Patent Document Model. The Patent Document Model (PDM) represents the information contained in a typical patent document and is based on the patent document structure defined by Lim and Kwon (2016), which depicts a common application format agreed

[2]https://www.w3.org/TR/DOM-Level-2-Core/ [last accessed December 10, 2023]
[3]https://www.omg.org/spec/UML/2.0 [last accessed December 10, 2023]

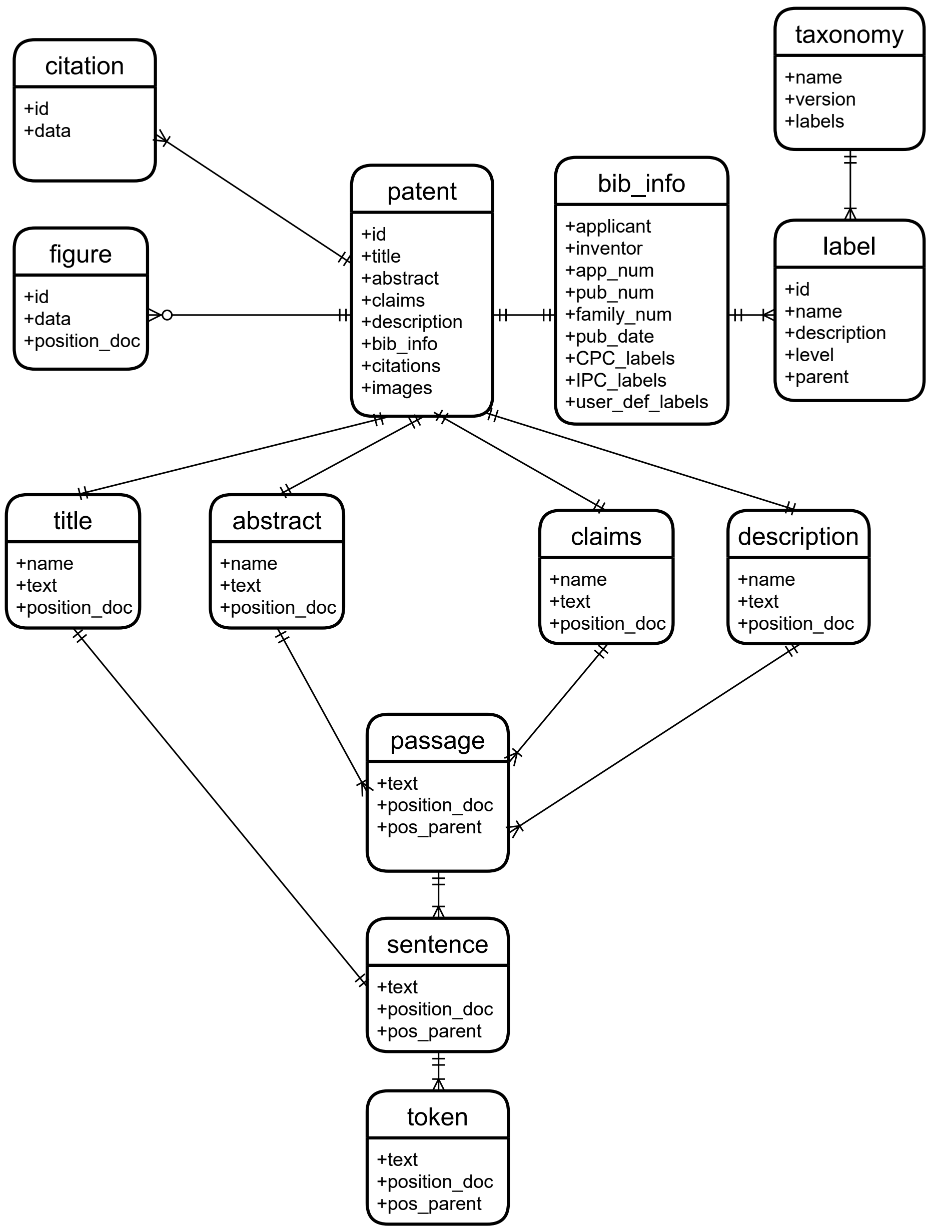


Figure 3.3: A patent document model showing different elements for a typical patent based on Lim and Kwon (2016).

upon by different patent offices.[4] The PDM represents the hierarchical structure of a patent document, with the patent entity at its root (Figure 3.3). It contains instances of title, abstract, claims, description, passage, sentence, token, figure, bib_info, and citation. The title, abstract, claims, description, passage, sentence, and token entities represent textual information. In contrast, the figure entity represents the figures within the patent. A figure description is associated with each of the figures. The bib_info entity represents

[4]https://www.epo.org/applying/online-services/online-filing/auxiliary/patxml/caf.html [last accessed December 10, 2023]

the patent metadata, and the citation entity represents the patented and non-patented prior art referenced within a patent.

The title entity represents the title of a patent, while the abstract, claims, and description entities represent the other three main sections of a patent. Each of these has name, text, and position_doc attributes. The name attribute represents the name of a section, text represents its textual content, and position_doc represents its position within the document relative to the other sections. Additionally, abstract, claims, and description contain instances of passage, sentence, and token entities that occur in the same order in the hierarchical document structure. Each of the three entities, passage, sentence, and token, has text, position_doc, and pos_parent attributes. An additional attribute, pos_parent, represents the position of an instance among its siblings of the same type.

In addition to textual content, a patent may contain figures depicting the details of the proposed system or process. These figures are represented as instances of the figure entity, which has id, position_doc, data, and fig-desc attributes. Id uniquely identifies a figure within the patent, position_doc represents its position relative to other figures in the patent, data represents the associated image data, whereas fig-desc represents a textual description of the figure.

Furthermore, a patent might refer to patents or non-patent prior art, represented as instances of the citation entity. The citation entity consists of two attributes, id and data, where id is a unique identifier within the document and data represents the metadata associated with cited work, e.g., title, author names, and publication date.

Above, we described the document model for a typical patent, which can be adapted for patent classification tasks and datasets. For example, the USPTO-70k dataset described in Section 4.3 consists of brief-summary, detail-desc, and fig-desc as additional fields, instead of description. We therefore adapt PDM by replacing the description field with three of its sub-fields. Similarly, a PDM for the PLS classification task consists of a list of PLS-oriented categories as an additional metadata attribute.

PubMed Document Model. To show the applicability of the proposed classification techniques to document types other than patents, we evaluate their application to the classification of PubMed articles (see Section 7.6). As discussed, the classification pipeline can be adapted for a specific use case by defining an appropriate DSDM and a corresponding mapper function, given that the document representation method and classification models are still valid for the new task setting.

When defining the PubMed Document Model (PubMedDM) based on the dataset provided for Task 5 of the BioCreative VII challenge (Chen et al., 2021b), we add title and abstract as content fields. Further, journal, keywords, and publication type (pub_type) are added as metadata attributes (Figure 3.4). The proposed document model can be further adapted for a task or dataset related to PubMed articles.

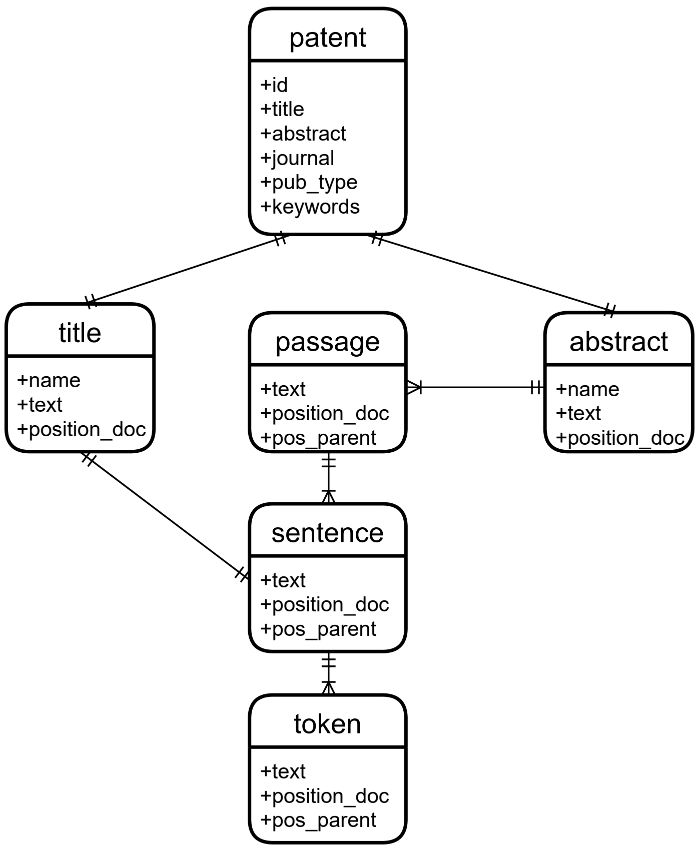


Figure 3.4: Document model for a typical PubMed instance within the LitCovid dataset (Chen et al., 2021c).

3.2.3 Domain-Independent Document Model

As discussed, DIDM represents the information within a document independent of domain and document type, such that once a DSDM instance is mapped to a DIDM instance, the remaining operations can be applied to it independent of its type. DIDM represents the common information observed in the templates of different document types. Looking through the DSDMs discussed in Section 3.2.2, we find that both consist of multiple content fields containing paragraphs, sentences, and tokens in a hierarchical structure. Further, multiple metadata elements are associated with a document. Based on the common content and structural patterns observed across document types, we propose the DIDM depicted in Figure 3.5 and described in the following paragraphs.

According to Ingwersen (1994), a document consists of semantic entities arranged at different levels of granularity, which can be aggregated into a representation. To avoid confusion with the term "entity" as used in NLP and knowledge management, we refer to these semantic entities as "semantic elements". Our DIDM defines the semantic element as the main entity type. The semantic element has seven attributes: id, id_doc, type, data, data_type, parent_id, and ml_feature.

The id attribute uniquely identifies a semantic element instance in the corpus, whereas id_doc identifies it uniquely within the document. The type attribute represents the type

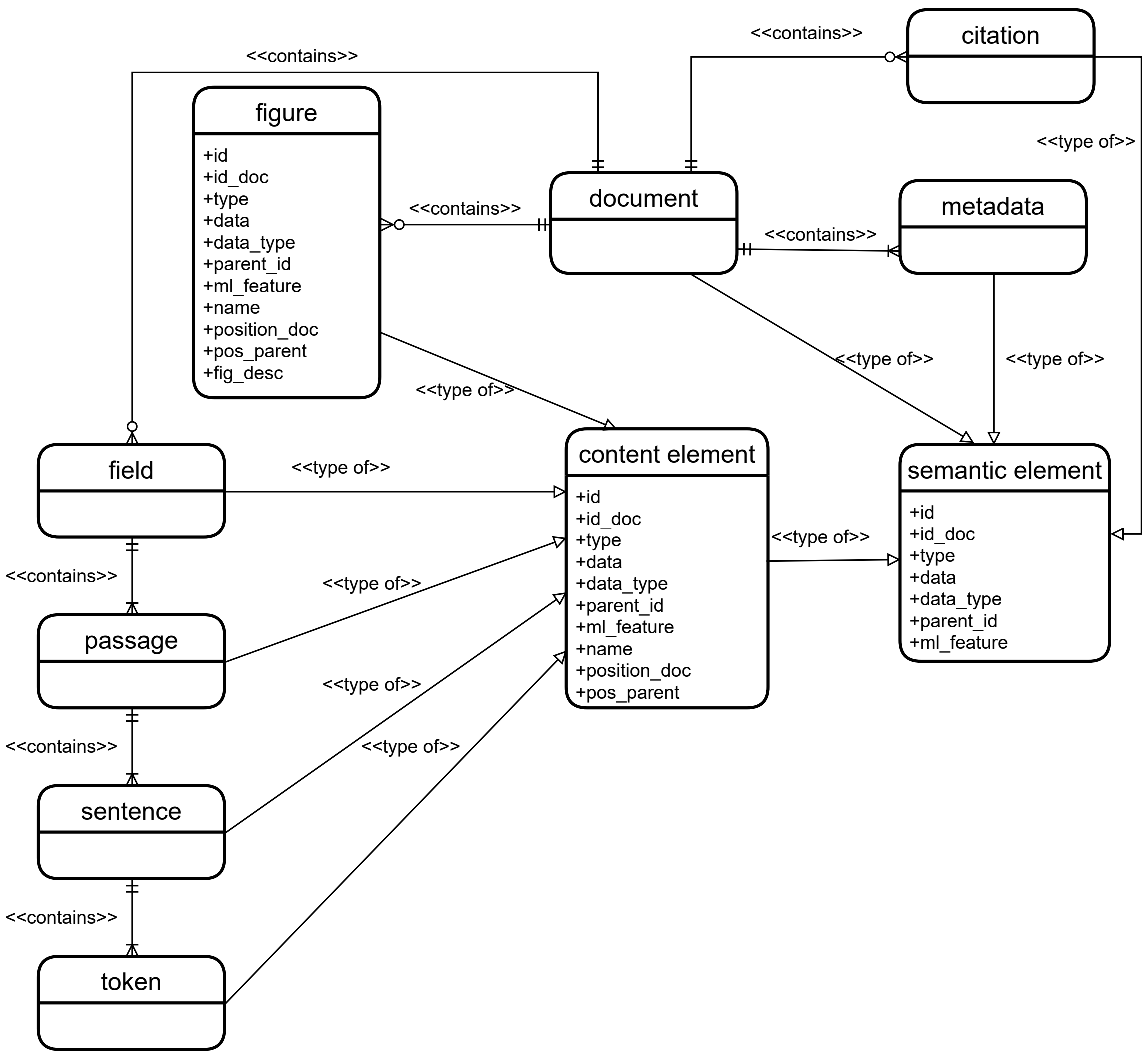


Figure 3.5: The Domain-Independent Document Model (DIDM) represents the key structural patterns observed across multiple document types, e.g., patents and research publications.

of a semantic element (defined below). The data attribute represents the associated data, and the corresponding data type information is represented by the data_type attribute. Finally, the information within a semantic element and its context can be embedded as a vector represented by the ml_feature attribute. In a non-neural classifier, the ml_feature attribute is referred to as the "feature vector", whereas in a neural network model, it is referred to as "embedding".

A document contains multiple instances of semantic elements arranged in a hierarchical structure, which occurs in a linear order within the document, information captured by position_doc and pos_parent attributes. The parent-child relationships between the semantic elements are captured with the parent_id attribute, which links a semantic element to its parent.

The DIDM categorizes semantic elements into four types: document, content element, metadata, or citation, as shown in Figure 3.5. The document type is at the root of the document tree and contains instances of field, figure, metadata, and citation as child nodes.

Instances of content element represent the document's content and have three additional attributes: name, position_doc, and pos_parent. The name attribute represents the name or title associated with the content element, while the position attributes, position_doc and pos_parent, denote the position of a content element instance relative to other instances of the same type in the document or to its siblings (i.e., child nodes of the same parent).

Although physically, the semantics elements are arranged sequentially, logically, it can be represented as a tree with document element at its root. The document text is structured with one or more sections and subsections containing passages, sentences, and tokens in a hierarchical order. Sections and subsections are represented with the field entity, while passages, sentences, and tokens are represented with the passage, sentence, and token entities, respectively. In addition to the content element types discussed above, a document might contain figures, represented as instances of the figure entity, with the additional attribute fig_desc.

Metadata, including labels, keywords, the publication date, and the authors' names, might be associated with a document to enable searching and management of the document corpus, which is represented as the metadata entity. Labels and keywords may indicate the main topic of a document and may, therefore, be relevant to classification tasks. Information about other documents referred to within the document being modeled is represented by the citation entity, which has the same attributes as the semantic element. Metadata for these other documents are provided by the data attribute in the citation entity.

$$\begin{aligned} U^c &= F \cup P \cup S \cup T \cup I \\ U &= U^c \cup M \cup Z \end{aligned} \tag{3.1}$$

We represent different entities within a document as a set of semantic elements U, containing instances of content element, metadata, and citation, represented as sets U^c, M, and Z, respectively (Equation 3.1). Furthermore, U^c contains instances of field, passage, sentence, token, and image, represented as sets F, P, S, T, and I, respectively.

3.2.4 Data Model for Representation Methods

According to Ingwersen (1994), a document is composed of semantic elements arranged at different levels of granularity. When generating a document representation, we can select a subset of semantic elements from set U and use them to represent a document. For example, we might create a document representation by using only a subset of sentences from the document. This can be done by creating an instance of DocRepDM, which represents the input for a document representation method. Here, we define a few of the DocRepDM types crucial for the document representation techniques discussed in the subsequent chapters.

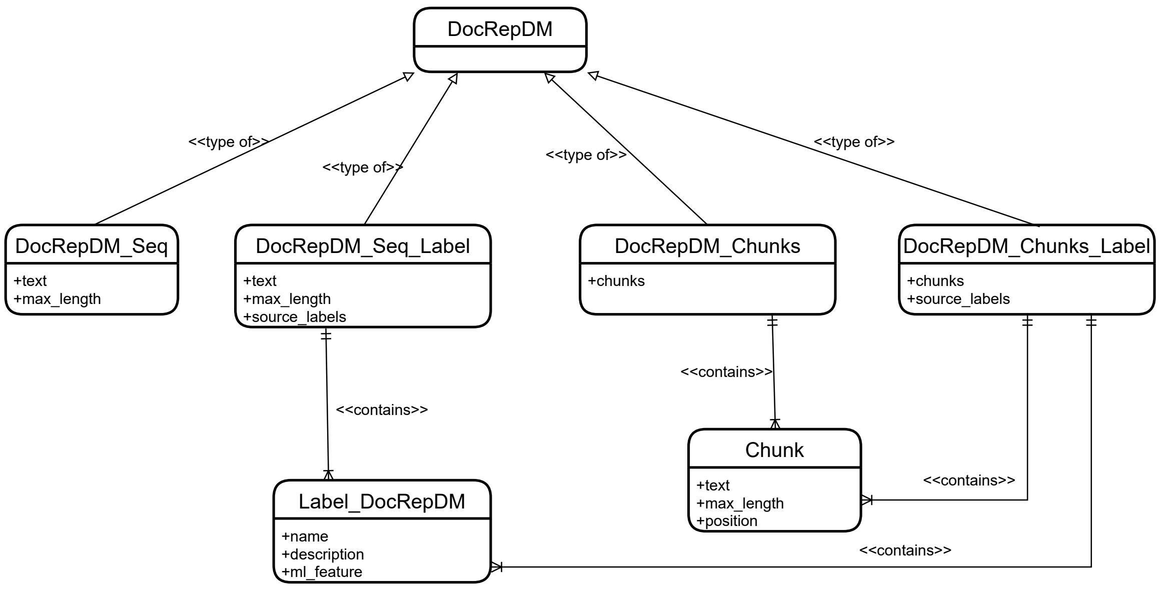


Figure 3.6: Different types of DocRepDM and associated data models.

The majority of text representation methods take a tokenized sequence as input. However, we must consider situations where a text representation method expects chunks of text instead of a sequence. For example, in Chapter 6, we experiment with text representation methods, which select top-k most informative sentences in a document based on a ranking method and then use them to generate a representation. We define two data models based on the document representation methods discussed, which are $\mathrm{DocRepDM\_Seq}$ and $\mathrm{DocRepDM\_Chunks}$.

The $\mathrm{DocRepDM\_Seq}$ data model represents the input to a document representation method that takes a sequence as input and has two attributes text and $\mathrm{max\_length}$ (Figure 3.6). The text attribute is the text to be tokenized, and the $\mathrm{max\_length}$ attribute corresponds to the maximum length of the tokenized sequence. The document representation methods used in Chapter 5, which generate a sequence representation using SciBERT, takes an instance of $\mathrm{DocRepDM\_Seq}$ as input (see Section 5.3).

$\mathrm{DocRepDM\_Chunks}$ represents the input to a document representation method that takes a list of text chunks as input. $\mathrm{DocRepDM\_Chunks}$ has chunks as only attribute, which is a list of Chunk instances (Figure 3.6). Chunk has text, $\mathrm{max\_length}$, and $\mathrm{position}$ as attributes, where text is the text associated with the text chunk, $\mathrm{max\_length}$ is the maximum allowed length for the tokenized sequence, and $\mathrm{position}$ corresponds to its position among the chunks in the text of the original document. The document representation techniques in Chapter 6 employing embeddings from texts corresponding to the selected fields and sentences are examples of methods that take an instance of $\mathrm{DocRepDM\_Chunks}$ as input (see Section 6.3).

In addition to the textual information, metadata may be associated with a document, which can be leveraged to generate a representation. For example, in the context of the PLS classification task, during inference, CPC/IPC labels are associated with a patent, which can be leveraged to predict PLS-oriented categories as "target labels" (see Definition 2.2, page 19). The CPC/IPC labels and PLS-oriented categories are associated with two different taxonomies, and we refer to CPC/IPC labels as "source labels". Similarly, for the task of classifying PubMed articles into COVID-19 categories, we can refer to author-provided keywords associated with a PubMed article as "source labels", which are known during inference (see Section 7.6).

Considering the use cases discussed above, we append source_labels attributes to DocRepDM_Seq and DocRepDM_Chunks to create DocRepDM_Seq_Label and DocRepDM_Chunks_Label, respectively. The source_labels attribute is a list of instances of type Label_DocRepDM, where the Label_DocRepDM data model has two attributes: text and ml_feature. The text represents the textual description of a source label instance, whereas ml_feature is its vector representation. Further types of DocRepDM can be added based on task setting and the associated document representation method. The document representation methods proposed for the PLS classification that combines embeddings for the textual fields with the CPC/IPC label embeddings takes an instance of DocRepDM_Chunks_Label as input (see Section 7.3). In contrast, the best-performing approach for the task of classifying PubMed articles to COVID-19 categories takes an instance of DocRepDM_Seq_Label as input (see Section 7.6).

In this chapter, we introduced a conceptual model for document classification. We expanded it further to define the classification pipeline that consists of components and document models that could be adapted for a use case or classifier. We defined domain-specific data models for patents and PubMed articles and drew commonalities across these two data models to define the domain-independent document model. Further, we proposed components and data models specific to the document representation methods. The conceptual model proposed in this chapter enables us to use the techniques defined in the later chapters in similar task settings with a few minor adaptations.

Chapter 4

Datasets and Analysis

This thesis aims to develop classification techniques for addressing the patent classification task in different application scenarios. However, the unavailability of adequate datasets is one of the major bottlenecks for the design and evaluation of classifiers targeting patent classification tasks. In this chapter, we address this challenge and present several datasets that have been made publicly available as important contributions to the research community (Sections 4.3 and 4.4). Further, we present in-depth analyses of our datasets in Section 4.5 and Section 4.6, which enhanced our understanding of their characteristics and potential challenges for different classifier setups.

The first analysis of the PLS classification datasets determined the association between CPC/IPC labels and PLS-oriented target categories. The output of the analysis was used to interpret results, assuming that for a dataset with a higher association, the document representation techniques using CPC/IPC information work better for the PLS classification task (see Definition 2.2, page 19). Chapter 7 validates this assumption with an in-depth evaluation. The second analysis revealed the duplicate nature of patents' texts. It motivated the document representation techniques proposed in Chapters 6 and 7, which use the full texts of patents for CPC/IPC and PLS classification tasks.

4.1 Motivation and Contributions

The unavailability of adequate datasets hinders the development of efficient and effective patent classification methods. This chapter introduces the novel datasets used in the later chapters to evaluate and compare various classification techniques that rely on different document representation methods and classification models. Finally, each dataset is analyzed to derive characteristics that support the selection of a suitable document representation technique for a certain task and the interpretation of model output.

Previous works tackling patent classification tasks with neural network models have primarily used the title, abstract, and claims as patent fields while leaving out the more elaborate description section due to the computational reasons (Li et al., 2018a; Lee and Hsiang, 2019; Althammer et al., 2021a). However, it is difficult to judge the importance of various sections and their impact on classification performance. To the best of our knowledge, no comprehensive study has yet been published on the role played by different patent sections and associated sentences when performing patent classification. To perform such a study and evaluate text representation methods over the full texts of patents, we curate and release the USPTO-70k dataset. This dataset contains the full texts of 70,000 patents and corresponding field information for each instance, along with the associated subclass labels (Section 4.3).

The PLS classification task differs from the CPC/IPC classification task (see Definition 2.1, page 2.1) such that a training instance contains an additional set of PLS-oriented categories as well as CPC/IPC labels. When testing the classification model with the test set, the CPC/IPC labels are known and can thus be incorporated into a document representation, whereas the unknown PLS-oriented categories are predicted as target labels. Manual labeling requires considerable time and effort from domain experts, and therefore, to the best of our knowledge, no dataset exists for the PLS classification task. Work by Richter and MacFarlane (2005) addressed a related task with patent alert generation as an application, but the dataset was not released as the algorithm was developed for a proprietary system. The lack of labeled datasets is a significant roadblock when developing classifiers for the PLS classification task. To address this issue, we release three PLS datasets from two diverse domains (Section 4.4).

The first dataset, InjVal, contains patent families labeled with categories describing types of injection valves and related technologies. The patents in the InjVal dataset were labeled by an in-house domain expert in Robert Bosch GmbH[1], a patent attorney with over 30 years of experience in injection valves and related IP management who has performed the classification task weekly for the past 25 years. The other two datasets are enriched versions of two smaller document collections from the World Intellectual Property Organization (WIPO), which were created during real-world PLSs on HIV drugs, namely Ritonavir[2] (Rito) and Atazanavir[3] (Atz) (Section 4.4).

The datasets were analyzed to derive dataset characteristics. There were two main reasons for this. First, the dataset characteristics help facilitate the interpretation of evaluation results for the proposed classification models and baselines. Second, certain dataset characteristics support the design and selection of specific document representation techniques. For example, as abstracts are written incautiously and are often duplicated across patents, we were motivated to include additional patent fields in the document representation. We performed a basic statistical analysis of the USPTO-70k and PLS datasets, followed by two detailed analyses.

[1]https://www.bosch.de/ [last accessed December 10, 2023]

[2]https://www.wipo.int/publications/en/details.jsp?id=230&plang=EN [last accessed December 10, 2023]

[3]https://www.wipo.int/publications/en/details.jsp?id=265&plang=EN [last accessed December 10, 2023]

The first analysis is relevant to the PLS classification task and determined the association between CPC/IPC labels and PLS-oriented target categories (Section 4.5). We hypothesized that a classifier exploiting CPC/IPC information would perform better for datasets with higher association between CPC/IPC labels and PLS-oriented categories. To test this, we calculated and analyzed Pointwise Mutual Information (PMI) (Church and Hanks, 1990) for each CPC/IPC label and PLS-oriented category pair for each of the three PLS datasets. Further, we evaluate the CPC/IPC-based document representation in Section 7.5.

The second analysis revealed the issue of duplicate text within and across documents (Section 4.6). We found that abstracts within the USPTO-7M dataset are often reused across patents, which motivated us to consider text fields other than just titles and abstracts when addressing patent classification tasks. Additionally, the same sentence may be reused multiple times within or across documents, indicating potential information redundancy in patent text. As a result, adding more text may not improve the performance of a classifier. Based on this finding, we hypothesize that a classifier can achieve optimal classification performance with limited but informative text elements. Chapter 6 builds on this hypothesis and proposes efficient document representation techniques that achieve optimal model accuracy with limited text elements (see Section 6.3).

Contributions. The main contributions described in this chapter are summarized as follows:

- We release the USPTO-70k dataset in which the **full text of the patent** is associated with each instance together with the corresponding field information (Section 4.3). This enables us to evaluate efficient document representation techniques that use the full texts of patents in Chapter 6.
- Furthermore, we release **three datasets from two diverse domains** (Section 4.4), based on which document representation methods are developed and evaluated in Chapter 7. This targets the third step of a PLS process, i.e., classifying patents into user-defined PLS-oriented categories.
- We provide a methodology for the determination of the association between CPC/IPC labels and PLS-oriented categories for a dataset (Section 4.5). This facilitates **the selection of suitable document representation techniques** for the PLS classification task.
- The **analysis of duplicate text**, which includes abstracts and sentences, shows that the information within a **patent document is redundant** (Section 4.6). This finding motivates document representation techniques (proposed in Chapter 6) that use **limited yet informative text elements**.

4.2 Related Datasets

Datasets for CPC/IPC Classification. WIPO-alpha is one of the first open-source datasets to have been used over the years for evaluating patent classifiers (Fall et al., 2003). The CLEF-IP 2011 challenge led to the release of a multilingual IPC classification dataset, which includes a test set divided into three parts containing 1000 instances each for three languages: English, German, and French (Piroi et al., 2011). The complete CLEF-IP dataset of 1.5M patents can be used as a training set. Targeting the patent classification for CPC taxonomy, Li et al. (2018a) propose a CNN-based classifier and trained it using a dataset containing 2M patents from the United States Patent and Trademark Office (USPTO), which they referred to as USPTO-2M. The USPTO-2M dataset contains training instances from 2006 to 2014, whereas the test set contains patents from 2015. Li et al. (2018a) evaluate classifiers using title and abstract as input. The USPTO-2M dataset was also used by Lee and Hsiang (2019) to evaluate a BERT-based classifier with title + abstract or first-claim as input. The USPTO-2M dataset does not contain the complete text of the patent documents and is, therefore, unsuitable when the goal is to experiment with the full texts of patents. Recently, Suzgun et al. (2022) released a patent dataset related to multiple natural language processing tasks, including CPC classification.

Datasets for Patent Landscape Studies. While some recent works have released datasets targeting the first step of the PLS process (Abood and Feltenberger, 2018; Choi et al., 2022), i.e., the document relevance classification task, to the best of our knowledge, no suitable datasets exist for training a patent classification model in the context of PLS classification task (see Definition 2.2, page 19). Giczy et al. (2022) released a dataset consisting of seed and anti-seed patents, which is used for exploring and filtering relevant candidate documents. Richter and MacFarlane (2005) study classification for a patent alert system in the biochemical domain, which is similar to classification in PLS, but the dataset was not publicly released.

Datasets for Patent Prior-art Search. During patent evaluation, examiners create search reports that map claims within an application to relevant passages in prior art. This information is used to determine the status of a patent application. For example, based on information from search reports Risch et al. (2020a) released a dataset that maps claims to prior art passages. The prior art search use case and the associated task were the focus of the patent retrieval task during the CLEF-IP 2011 challenge (Piroi et al., 2011), and the dataset released at that time has been used in many subsequent works on patent retrieval (Magdy and Jones, 2011; Andersson et al., 2016; Hofstätter et al., 2019; Althammer et al., 2021b). Kang et al. (2020) provide a detailed survey of recent patent prior art search methods.

Datasets for Patent Summarization. Sharma et al. (2019) provide a patent summarization dataset that summarizes the description section with the abstract as the target text. However, since abstracts are less carefully written and are often duplicated (Section 4.6), the use of abstracts as patent summary texts might not provide sufficient information for classification.

Summary. Looking through recent works relevant to CPC classification, we find that they use datasets that contain limited patent text and are unable to answer the crucial question of whether the performance of a neural network classifier can be improved with additional text (Li et al., 2018a; Lee and Hsiang, 2019; Althammer et al., 2021a). With this thesis, we release the USPTO-70k dataset, which contains patent instances with the full texts of patents (Section 4.3). The lack of adequate datasets is also a blocker to the development of multi-label classification techniques in the context of PLS. To enable the automation of PLS, we release three datasets targeting PLS classification (Section 4.4).

4.3 USPTO-70k

As discussed in the previous section, recent neural methods have been evaluated using limited patent text, and the impact of additional text on the performance of classifiers is not well studied. Therefore, to enable research on the development of efficient patent classification methods using the full texts of patents, we curate and release the USPTO-70k dataset sampled from the USPTO data dump of 7M patents. Unlike previous CPC classification benchmark datasets (Li et al., 2018a; Lee and Hsiang, 2019), USPTO-70k contains the full texts of patents together with the corresponding subclass labels for each patent instance. This allows us to experiment with different text element types, e.g., sections, passages, sentences, and tokens (see Section 6.5).

The USPTO data dump is an export of a relational database and includes tables corresponding to different patent fields. As of 01/04/2020, the USPTO data dump contained approximately 7M patents. The complete dataset is referred to as USPTO-7M. In the following, we provide details of the database tables for the USPTO data dump, information on the USPTO-70k dataset creation process, and the details of statistical analyses for patent text and CPC labels.

4.3.1 USPTO Data Dump

The United States Patent and Trademark Office (USPTO) provides open-source access to patent grants through the USPTO bulk dataset.[4] The bulk dataset is an export of a MySQL database in which the data is normalized into multiple tables, with patent being the main table as shown in Figure 4.1. The MySQL documentation[5] describes the key datatypes in MySQL: varchar, mediumtext, int, and date. A row within the patent table stores bibliographic information corresponding to a unique patent document. All other tables link to

[4]https://patentsview.org/download/data-download-tables [last accessed December 10, 2023]
[5]https://dev.mysql.com/doc/refman/8.0/en/data-types.html[last accessed December 10, 2023]

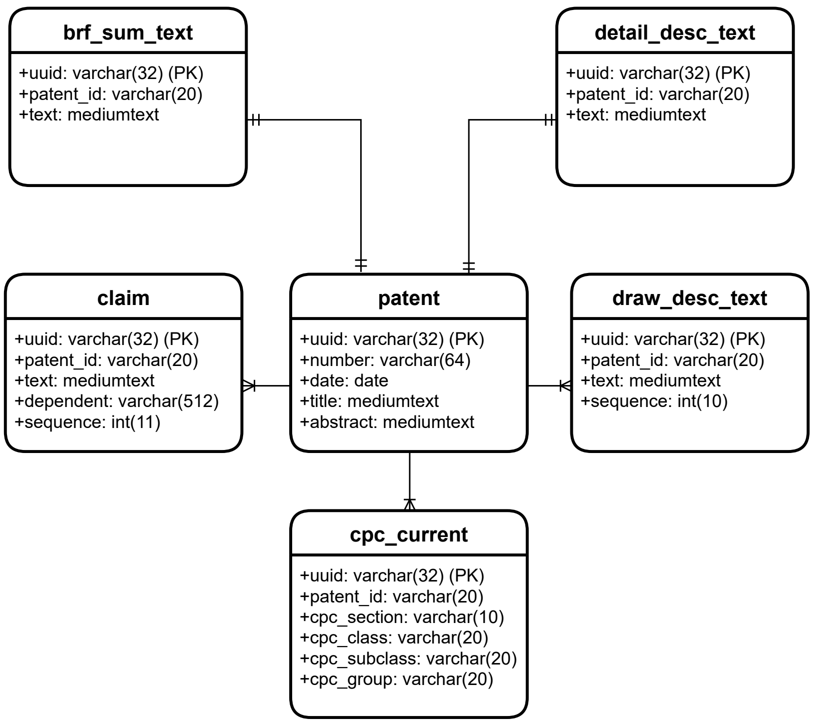


Figure 4.1: Entity-relationship diagram for USPTO bulk dataset showing different tables. Patent is the main table; other tables link to it through patent_id as a foreign key attribute.

patent using patent_id as a foreign key. USPTO periodically releases a new version to add recent patent grants to the data dump.[6]

Patent Text. Representing the bibliographic information for a patent document, the patent table contains five main attributes: id, number, date, title, and abstract. A patent can be uniquely identified with either an id or a number attribute, where the id is a randomly generated unique string, and the number corresponds to the publication number of the patent. The date attribute specifies the date on which a patent was granted. The title and abstract attributes represent information corresponding to the title and abstract of a patent, respectively.

Each claim within a patent document is stored as a separate row in the claim table, which contains patent_id, text, dependent, and sequence as its four main attributes. As a foreign key attribute, patent_id links a claim to the corresponding patent in the patent table. Claims within a patent occur in a particular sequence, and the sequence information is captured by the integer-valued sequence attribute in the claim table. As an ordered pair, the patent_id and sequence attributes uniquely identify a claim in the claim table. As discussed in Section 2.1.2, a claim can be one of two types: a dependent claim or an independent claim. For a dependent claim, the dependent attribute stores a list of claims

[6]https://patentsview.org/release-notes [last accessed December 10, 2023]

on which the given claim depends, whereas the value for the dependent attribute is NULL in the case of an independent claim.

The textual data from the description section is divided into three tables: brf_sum_text, draw_desc_text, and detail_desc_text. The details for each of the three tables are as follows:

- The brf_sum_text table stores textual information for subsections that are usually placed at the beginning of a description section, e.g., the field of invention, invention summary, and background. It contains patent_id and text as the two main attributes. As a foreign key, the patent_id attribute links a row to a specific patent in the patent table. The text attribute represents the textual data corresponding to the brief-summary subsection.
- Patents often contain drawings that are accompanied by a textual description. The draw_desc_text table stores textual descriptions of the drawings associated with a patent and contains patent_id, text, and sequence as its three main attributes. As a foreign key, the patent_id links the drawing description to a patent in the patent table. The text attribute represents the textual data for a drawing description. As drawings within a patent are positioned in a specific sequence, the sequence information is represented as the sequence attribute in the draw_desc_text table. Similar to the claims table, a row in the draw_desc_text table can be uniquely identified by a combination of patent_id and sequence attributes.
- With patent_id and text as its two main attributes, the detail_desc_text table stores textual data found within the "Detailed Description" subsection.

CPC Labels. The cpc_current table stores information about the CPC labels for a patent in the patent table. When assigning labels to a patent, the usual practice is for an examiner to select labels from the fifth level and below, i.e., subgroup labels. The cpc_current table contains patent_id, cpc_section, cpc_class, cpc_subclass, and cpc_group as attributes. As multiple CPC labels can be associated with a patent document, cpc_current divides the CPC label information into multiple rows. When retrieving the CPC labels corresponding to a patent, the cpc_current rows are aggregated for the corresponding patent_id.

4.3.2 Dataset Creation Process

USPTO-70k is sampled from the USPTO-7M dataset using our Python implementation, which predominantly relies on Pandas.[7] Pandas contains data structures for representing tabular data and provides methods with similar functionality to standard SQL operations[8], e.g., SELECT or GROUP BY. Figure 4.2 depicts the step-by-step process described in the following paragraphs.

[7]https://pandas.pydata.org/ [last accessed December 10, 2023]

[8]https://pandas.pydata.org/docs/getting_started/comparison/comparison_with_sql.html [last accessed December 10, 2023]

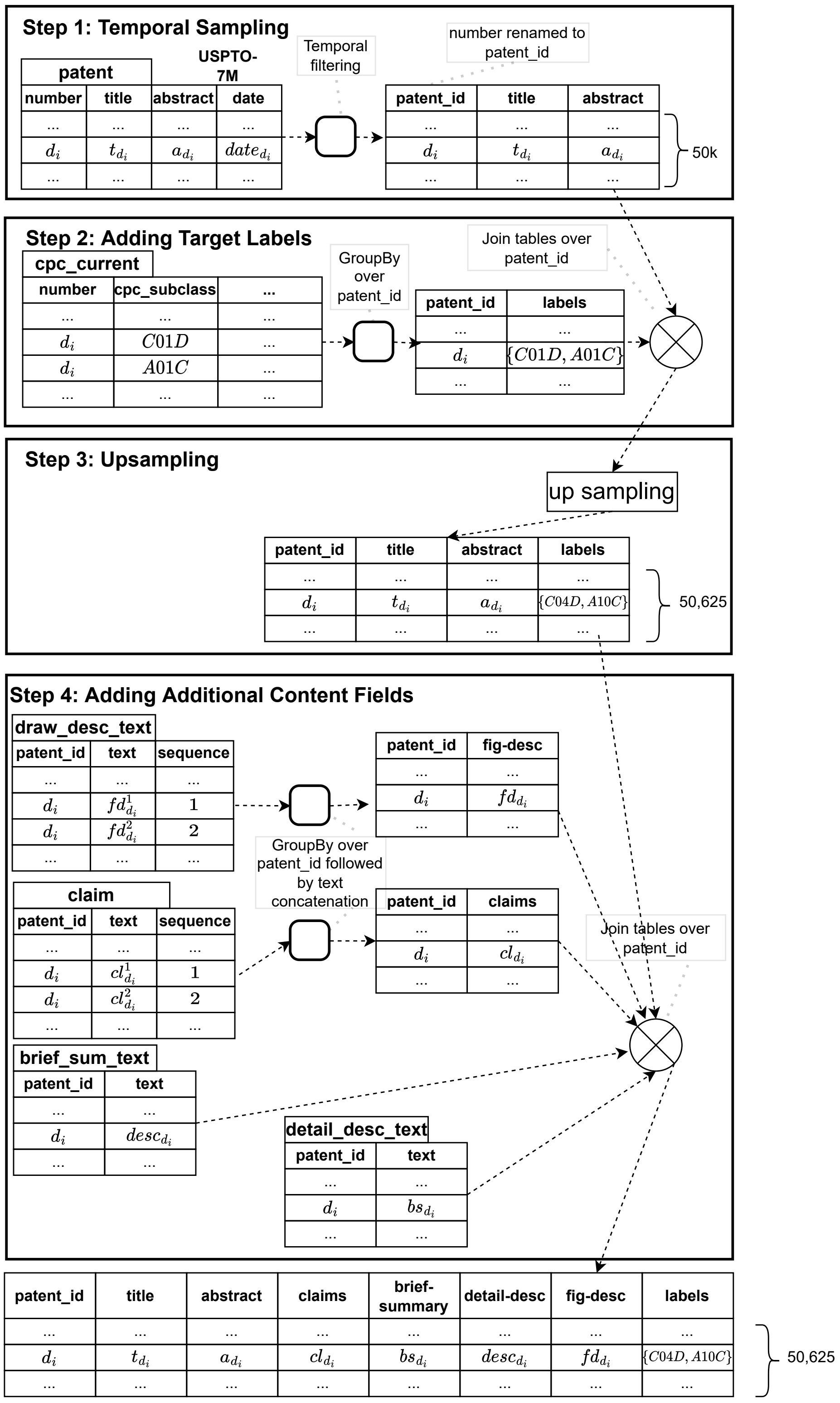

Step 1: Temporal Sampling
patent
USPTO-7M
number | title | abstract | date
Temporal filtering
number renamed to patent_id
patent_id | title | abstract
50k
Step 2: Adding Target Labels
cpc_current
number | cpc_subclass
C01D
A01C
GroupBy over patent_id
Join tables over patent_id
patent_id | labels
{C01D, A01C}
Step 3: Upsampling
up sampling
patent_id | title | abstract | labels
{C04D, A10C}
50,625
Step 4: Adding Additional Content Fields
draw_desc_text
patent_id | text | sequence
patent_id | fig-desc
GroupBy over patent_id followed by text concatenation
claim
patent_id | claims
Join tables over patent_id
brief_sum_text
detail_desc_text
patent_id | title | abstract | claims | brief-summary | detail-desc | fig-desc | labels
50,625

Figure 4.2: The process for creating the training split for the USPTO-70k dataset. A similar process is followed to create the validation and test splits, in which case the oversampling step (i.e., Step 3) is skipped.

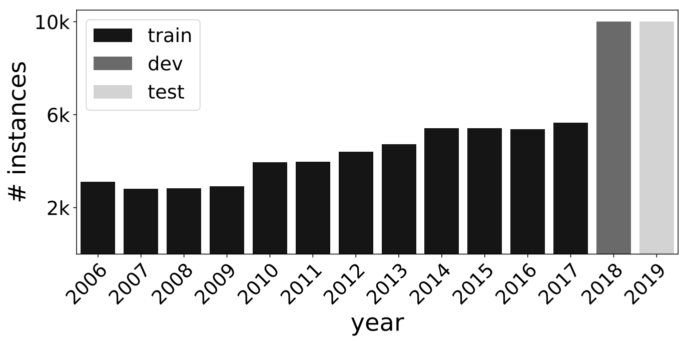


Figure 4.3: Distribution by year for the USPTO-70k dataset. The patents in the training set are from the years 2006 to 2017, the validation set contains instances from 2018, and the test set contains instances from 2019.

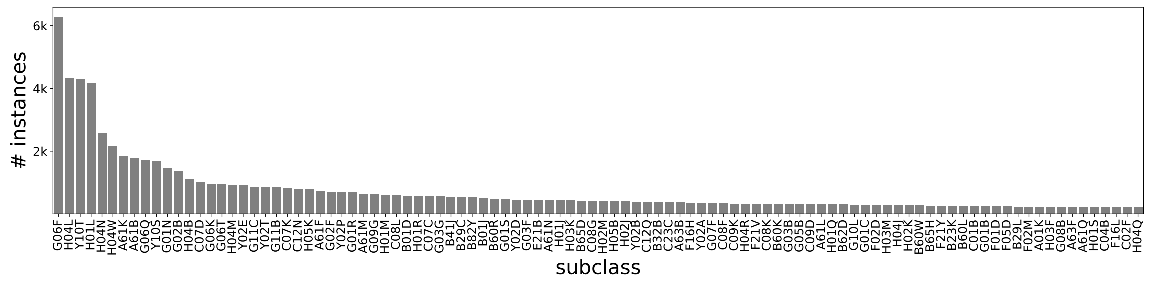


Figure 4.4: CPC distribution for the top 100 CPC labels in the USPTO-70k dataset.

Step 1: Temporal Sampling. A temporal dataset split creates a realistic setup in which models predict labels for newer patents based on older data, similar to the approach taken by D'hondt et al. (2014). This process assigns documents from 2006–2017 to the training set, 2018 to the validation set, and 2019 to the test set. The training set contains 50,000 patents, while the validation and test sets contain 10,000 patents each, for a total count of 70,000 (Figure 4.3). Next, the rows from the patent table are filtered using the 70k patent_id values. The output of Step 1 contains three columns corresponding to patent_id, title, and abstract.

Step 2: Adding a Target Column. Each row selected in Step 1 is accompanied by a unique patent_id value. Using the patent_id values, Step 2 filters the corresponding rows from the cpc_current table. Since the CPC information for a patent is distributed across multiple rows, the CPC labels corresponding to each patent are determined by performing a GROUP BY operation on the patent_id attribute for the filtered rows. The CPC classification techniques proposed in this thesis are evaluated for CPC labels up to the third level of the CPC taxonomy. Therefore, we determine CPC labels for patents up to the third level, i.e., subclass labels, and add them as the "labels" column in the tables corresponding to the training, validation, and test sets. The labels within the "labels" column represent the target labels for the CPC classification task.

	total labels			average labels per patent		
level	1	2	3	1	2	3
training	9	128	630	1.49	1.69	1.98
validation	9	126	575	1.56	1.84	2.25
test	9	127	573	1.56	1.89	2.32

Table 4.1: CPC label statistics for the USPTO-70k dataset showing the total labels and an average number of labels per instance for three hierarchical levels: section, class, and subclass, represented as 1, 2, and 3, respectively, in the table.

Step 3: Oversampling. The CPC labels suffer from a label sparsity problem, with many infrequent labels situated in the long tail of the CPC distribution; Figure 4.4 shows the distribution of the top 100 CPCs. The label sparsity problem is addressed by oversampling the least frequent labels. This is achieved by adding patents carrying infrequent labels, such that each label occurs at least ten times in the training set. After oversampling, all labels have at least ten instances in the training set. The oversampling operation is performed on the training set, adding 625 more instances. The validation and test splits are not changed, and thus, the initial distribution is kept the same. As shown in Table 4.1, the total number of labels at the leaf node, i.e., the subclass level, is 630 for the training set, 575 for the validation set, and 573 for the test set.

Step 4: Adding Additional Content Fields. The first three steps generate tabular data with four columns, and two of these columns correspond to the title and the abstract as the only content fields. Such a dataset is sufficient for training a classifier that takes the title and abstract as input. However, with this work, we aim to generate classification methods for long multi-field documents, for which we need a dataset that provides the full texts of patents and the corresponding field information. Therefore, in Step 4, we enrich the output of Step 3 with the content from four USPTO tables: claim, detail_desc_text, draw_desc_text, and brf_sum_text.

A row in the detail_desc_text and brf_sum_text tables corresponds to a unique patent. Since both tables contain information for all 7M patents, we first filter the relevant rows and then JOIN them to the Step 3 output using the patent_id attribute.

The claim and draw_desc_text tables distribute a patent text across multiple rows. To merge information from these tables, the relevant rows are first filtered and grouped using the patent_id attribute. Next, a two-column table is created with a patent_id as the first and second columns, containing the concatenated text from the rows for each of the patent_id groups. Such a table is created for both claim and draw_desc_text and merged to the Step 3 output with a JOIN operation on patent_id. The final output is an eight-column table saved in an easy-to-use Comma-Separated Variable (CSV) format. We described the patent document model (PDM) in Section 3.2.2 that represents the structure of a typical patent. PDM can be extended for the USPTO-70k dataset by adding entities corresponding to the subsections of description. We can initialize the patent instances using the extended PDM and USPTO-70k.

4.3.3 Corpus Statistics

This section describes the statistical characteristics of patent documents with regard to the distributions associated with patent texts and target labels for the USPTO-70k dataset. Further, labels are categorized into groups based on the use of taxonomy levels, broad technological fields, and label frequency as grouping criteria. Later chapters evaluate classifier performance over different label groups corresponding to three group types (see Section 6.3). Later in this section, we describe the token- and sentence-level distributions of patent texts, which reveal that patents are generally quite long and show significant variation in document length.

Label Statistics. There is one label occurring in the validation set that does not have any associated training instances. In the test split, there are seven such labels. The CPC statistics are shown in Table 4.1. The average number of labels per patent is around 1.5 at the first level of the hierarchy and up to 2.32 at the leaf level, with the latter increasing from 1.8 in 2014 to 2.3 in 2019.

Label Grouping. The following label groupings are created to better understand a classification model's performance across different dimensions, such as label frequency, taxonomy levels, and topics.

Grouping by Label Frequency. The less frequent labels typically capture fine granular information and are thus often more informative than the more frequent labels. Previous works have evaluated the performance of models for these less frequent labels in a few-shot setting. However, there is no standard threshold for what constitutes a minority class in few-shot text classification. For example, MIMIC-III (Johnson et al., 2016), EURLEX57K (Chalkidis et al., 2019), and AMAZON13K (Lewis et al., 2004) have few-shot categorization thresholds of 5, 50 and 100, respectively. Therefore, instead of sticking to a single value as a measure of a few-shot category, four frequency-based label groups are defined with a label frequency range of 0 to 10, 11 to 50, 51 to 100, and 101 or more, respectively. Frequency-based label groups are used in Section 6.5 to compare the performance of proposed document representations to baselines in a few-shot setting. The top-most part of Table 4.2 shows the number of labels within each group.

Grouping by Level. Hierarchy-based groups are created by using the hierarchical taxonomy information to divide the labels into groups based on the hierarchical level at which they appear. Since the USPTO-70k dataset contains labels from the first three levels of the CPC taxonomy, three label-based groups are created with 9, 128, and 630 labels, respectively (see the middle part of Table 4.2).

Grouping by Technical Field. Within the CPC taxonomy, the highest-level labels are referred to as section labels and represent nine topics at the highest granularity. To evaluate

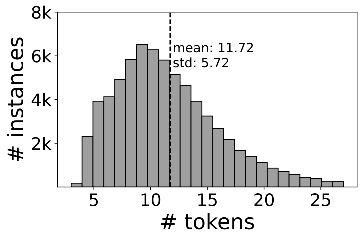
Grouping Criterion	**Grouping Identifier**	**Definition**	**Count (subclass) total=630**	**# Instances**
Label Frequency	1-10	$f_i \leq 10$	114	1,085
	11-50	$10 < f_i \leq 50$	238	5,079
	51-100	$50 < f_i \leq 100$	91	5,897
	100+	$100 < f_i$	187	47,486
			Count (all labels) total=767	
Level	1	Level 1 CPC taxonomy	9	50,625
	2	Level 2 CPC taxonomy	128	50,625
	3	Level 3 CPC taxonomy	630	50,625
			Count (all labels) total=767	
Section	A	Human Necessities	100	8,157
	B	Performing Operations; Transporting	199	18,167
	C	Chemistry; Metallurgy	103	10,474
	D	Textiles; Paper	44	6,791
	E	Fixed Constructions	38	2,390
	F	Mechanical Engineering; Lightning; Heating; Weapons; Blasting	119	15,591
	G	Physics	93	20,458
	H	Electricity	56	20,206
	Y	General Tagging	15	9,360

Table 4.2: Distribution of labels within three label groups for the USPTO-70k dataset.

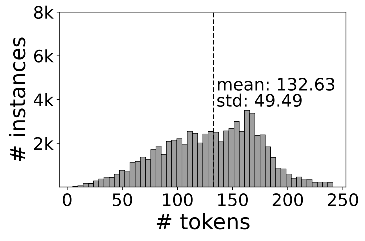
the performance of a classifier on different topics, the CPC labels are grouped into nine technical groups, where a group contains all the subclass labels as an indirect child of a section label (see bottom-most part of Table 4.2). When comparing different topic groups, we find that Group B contains the largest number of labels, followed by Group F. Group Y comprises only 15 general tagging labels for new technologies or general tagging labels for cross-sectional technologies.

For the USPTO-70k dataset, texts from patent fields are analyzed at the sentence and token levels. In the following paragraphs, we provide detailed analyses of the distribution of sentences and tokens.

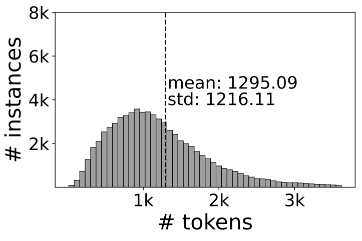
Token Distribution. The field text is tokenized to a sequence of word-piece tokens using the SciBERT tokenizer (Beltagy et al., 2019). Figure 4.5 shows the count of word-piece tokens in different fields. The title contains very few tokens, followed by the abstract and fig-desc. The claims field contains more word-piece tokens than the title, abstract



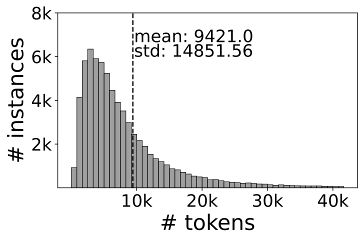
(a) title



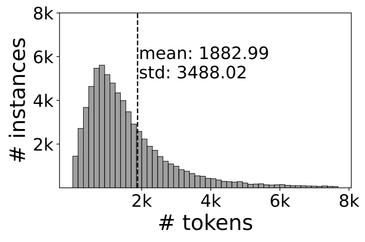
(b) abstract



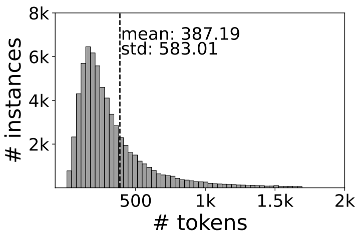
(c) claims



(d) detail-desc



(e) brief-summary

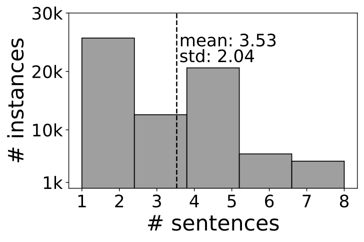


(f) fig-desc

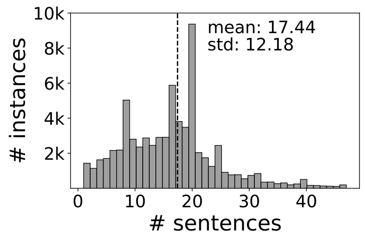
Figure 4.5: Token distribution for different patent fields in the USPTO-70k dataset.

field-name	**training-set**	**validation-set**	**test-set**
total number of instances	50,625	10,000	10,000
invention summary missing	5,836	1,428	1,413
background of invention missing	3,888	845	855
technical-field missing	28,304	5,111	5,008

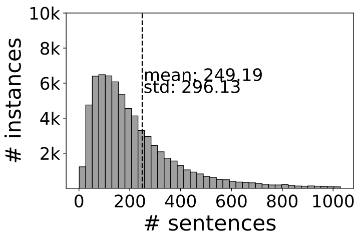
Table 4.3: Statistics showing the number of patents for which the subfields of brief-summary are missing in USPTO-70k.



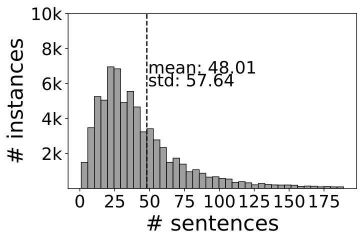
(a) abstract



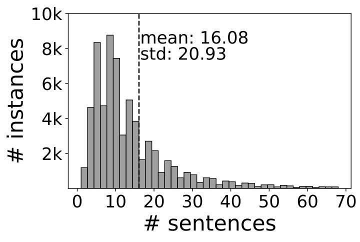
(b) claims



(c) detail-desc



(d) brief-summary



(e) fig-desc

Figure 4.6: Sentence distribution for different patent fields in the USPTO-70k dataset.

or fig-desc fields. With approximately 1.8k word-piece tokens, brief-summary is very concise compared to the elaborate detail-desc field, which has approximately 9.5k tokens on average (Figure 4.5). The brief-summary field contains several subfields. The statistics in Table 4.3 show that these subfields are not present in all the patents. Therefore, we concatenate the corresponding text instead of considering the subfields separately.

Sentence Distribution. A document representation can be generated by aggregating information at the sentence level; when doing so, it is important to know the distribution of sentences within a field and the document. As a pre-processing step, each patent field is tokenized using the NLTK sentence tokenizer.[9] Figure 4.6 shows the sentence distribution for different patent fields. Similar to the token statistics, the detail-desc is found to be the longest field with a mean sentence count of 249, whereas abstract is the shortest field containing 3.5 sentences on average. When aggregating the average number of sentences across fields, we find that a patent contains more than 300 sentences on average.

4.3.4 Summary for USPTO-70k

We provide details of the step-by-step process for creating the USPTO-70k dataset (Section 4.3.2), followed by analyses of CPC labels and patent text (Section 4.3.3). The USPTO-70k dataset was created from the larger USPTO data dump (described in Section 4.3.1). Each instance within the USPTO-70k dataset contains the full text of a patent and the corresponding field information and is labeled with a set of subclass labels. Analysis of label frequency reveals a skewed label distribution in which a large number of labels are associated with very few patents. Based on this observation, CPC labels are grouped into four frequency-based label groups that assist the comparison of different document representation techniques and classification models for various frequency-based groups (see Section 6.5). Further, our analysis of tokens and sentences shows that the patent documents are very long and vary widely in document length. The next section provides information on the dataset creation process for Patent Landscape Study datasets.

4.4 Patent Landscaping Datasets

In contrast to the CPC/IPC classification task, the PLS classification task expects the training instances to be labeled with labels taken from the CPC/IPC taxonomy (see also Definition 2.2, page 19). As such labels are likely to be informative, they can be incorporated into a document representation to predict the labels in the target taxonomy. This section introduces three new datasets, Injection Valve (InjVal), Ritonavir[10] (Rito), and Atazanavir[11]

[9] https://www.nltk.org/api/nltk.tokenize.html [last accessed December 10, 2023]

[10] https://www.wipo.int/publications/en/details.jsp?id=230&plang=EN [last accessed December 10, 2023]

[11] https://www.wipo.int/publications/en/details.jsp?id=265&plang=EN [last accessed December 10, 2023]

(Atz), for the PLS classification task. These datasets are taken from two diverse domains, namely mechanical systems (InjVal) and biochemistry (Rito and Atz). Detailed descriptions for each of the datasets are provided in the following sections.

4.4.1 Injection Valves Dataset

In the InjVal dataset, patent families are labeled with categories describing types of injection valves and related technologies. The dataset was provided by an industrial collaborator; a domain expert working for Robert Bosch GmbH[12] has performed the patent classification process weekly for the past 25 years. Each week, a candidate set of patents is generated by an alert system that filters new incoming patents using a CPC-based search query. The domain expert identifies relevant patents in the candidate set and categorizes them into a technical target category. Since most of these patents are for mechanical systems, the domain expert often refers to the figures while determining a patent's relevance to a technical category.

The 9,465 patent families are labeled with 16 different target labels, e.g., "Exhaust Line Injector", "Water Injection", indicating which injector components or injection types are used in the patents. Most of these patents are from the Japanese and German Patent Offices, but the dataset also includes US patents (Figure 4.9a). The corresponding English machine-translated text for each field is added using PatBase API.[13] The dataset covers a broad domain (5,068 CPC labels) and corresponds to a broad-scope PLS.

4.4.2 Ritonavir and Atazanavir

These two labeled datasets are derived from two publicly available PLSs that were performed by the World Intellectual Property Organization (WIPO) on Ritonavir (Rito) and Atazanavir (Atz), two drugs developed for the treatment of HIV infections and AIDS. The motivation for these studies, both conducted in 2011, was to track the development of the drug manufacturing process as well as the compositions and usage of these drugs since the filing of the first patents. In contrast to InjVal, these two datasets contain patent families within the narrow scope of a patent.

Dataset Creation Process. The WIPO studies were conducted iteratively. First, a keyword-based search yielded a list of relevant patents, which was then further refined using relevant CPC labels. Using a forward-backward citation search, some additional patents were identified and added to the dataset. Each study provides a spreadsheet-like overview with meta-information about the search and patents. For instance, labels assigned to the patents can be used to train a classifier for the PLS classification task. These labels have been carefully assigned by WIPO professionals during search (for Rito) and post-hoc with the support

[12]https://www.bosch.de/ [last accssed 10 December, 2023]
[13]https://www.patbase.com [last accessed December 10, 2023]

Dataset	# Instances	# Unique Labels	Avg. # Labels Per Instance
InjVal	9,465	16	1.01
Rito	781	7	1.35
Atz	640	8	2.14

Table 4.4: Target label statistics of PLS datasets.

of text mining software[14] (in the case of Atz). By analyzing the descriptions within the reports, we select subsets of these labels as PLS target labels for our experimental studies.

The main goal of the WIPO studies was to provide an article-style report, but the underlying data has been released as a spreadsheet. The Atz data, as provided by WIPO, contains the title and an abstract by Derwent[15], together with the first claim. The Rito data only lists title, abstract, and claims. As part of our contribution, a structured full-text dataset is derived from the information provided by WIPO by adding additional information from PatBase[16], which is released in an easy-to-use Comma-Separated Values (CSV) format. The full-text dataset includes the title, abstract, (all) claims, description text, CPC labels, patent number, family number, and publication date information.

Target Labels. The Rito dataset consists of 781 patents labeled with seven distinct target labels. These correspond to broad categories that have been assigned during a search by carefully choosing queries based on the combination of keywords (such as disease names or chemical compositions) with CPC classes. These categories include *Methods of Treating HIV*, *Combination*, and *Prodrug*, which relate to methods of administering the drug. The remaining four categories (*Pharmaceutical Composition*, *Derivatives, Synthesis, and Crystalline Forms*, and *Stabilized Forms*) define the form, composition, and derivatives of Ritonavir.

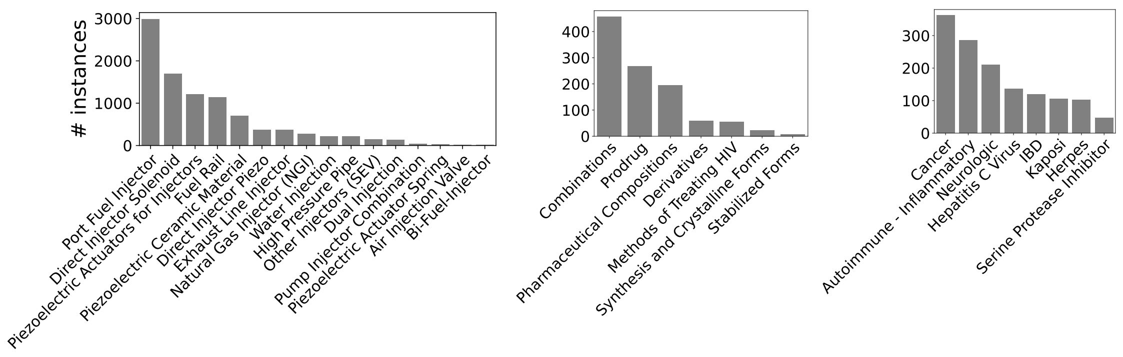
The Atz dataset consists of 640 patents and eight target labels, which are the names of the types of disease for which treatment is described in the patent, e.g., *Cancer*, *Kaposi*, and *Herpes*. While HIV is the primary indication for Atazanavir, medical professionals have administered the drug for other indications, and we select the subset of patents describing non-HIV indications, defining the target task as identifying the corresponding (non-HIV) disease. Among the target labels, *Cancer* is the most frequent, followed by *Autoimmune-Inflammatory*.

While both Atz and Rito focus on HIV-related drugs, the two PLS tasks are qualitatively different: Rito divides patents by technology, whereas Atz divides patents by application. As shown in Table 4.4, the average label per instance value is greater than 1, indicating a multi-label classification setting. The number of target classes for InjVal is twice that of the

[14]https://www.thevantagepoint.com/6-products/thomson-data-analyzer.html [last accessed December 10, 2023]

[15]https://www.clarivate.com/derwent/solutions/derwent-world-patent-index-dwpi [last accessed December 10, 2023]

[16]https://www.patbase.com [last accessed December 10, 2023]



(a) Injection-Valve **(b)** Ritonavir **(c)** Atazanavir

Figure 4.7: Label distributions of PLS-oriented categories in three PLS datasets.

Field	**InjVal**	**Rito**	**Atz**
abstract	104 ± 43	60 ± 35	56 ± 34
claims	358 ± 346	1215 ± 1051	1231 ± 948
description	2121 ± 1171	11579 ± 8800	16401 ± 10245

Table 4.5: Token counts for PLS datasets: Mean and standard deviation by dataset and textual field.

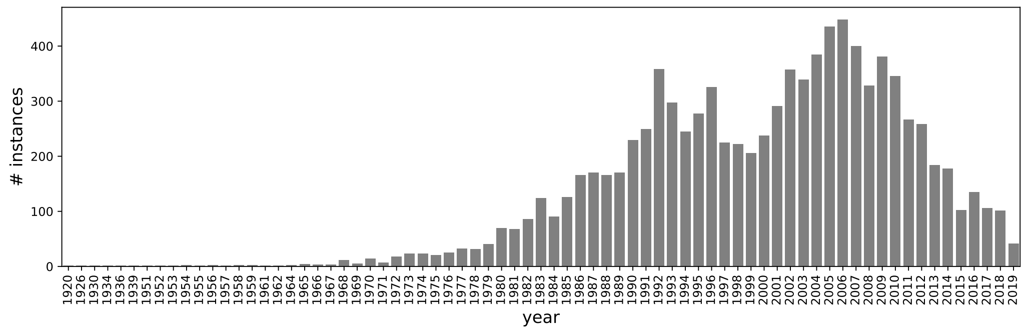
Rito and Atz datasets, and label distributions are highly skewed in all three datasets (see Figure 4.7). As shown in Figure 4.7, Atz is more balanced than the Rito dataset, where the most infrequent label in Rito has just seven instances.

4.4.3 Corpus Statistics

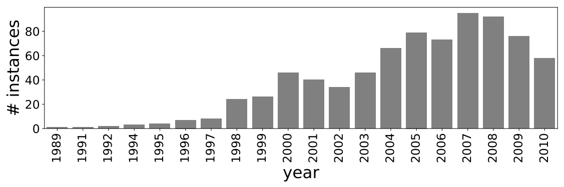
The characteristics of a dataset naturally affect the performance of classification models. To allow for a better interpretation of our experimental results, a statistical analysis of the datasets is performed.

Token Counts. The text for all patent fields is tokenized using the NLTK whitespace tokenizer, and average token counts are shown in Table 4.5. The abstracts in InjVal are longer compared to those of Rito and Atz, which have longer claims and description sections. High variation can be seen in the token count, particularly within the description sections for Rito and Atz.

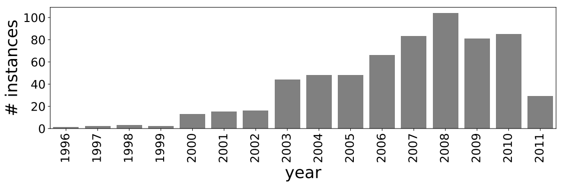
Publication Date. The publication date of a patent family is the earliest publication date among its family members. Figure 4.8 shows the time range for the three PLS datasets (InjVal, Rito, and Atz) using the publication date of the earliest family member. The patents within the InjVal dataset (Figure 4.8a) are spread across approximately 100 years (1920 - 2019), with a majority of them being from the last 50 years. In contrast, the WIPO datasets



(a) Injection-Valve

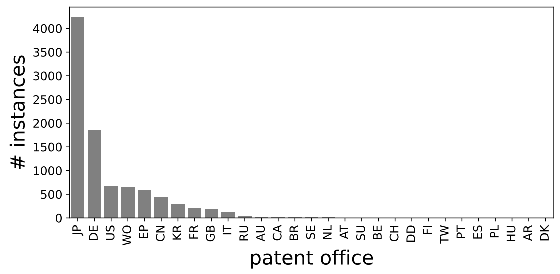


(b) Ritonavir

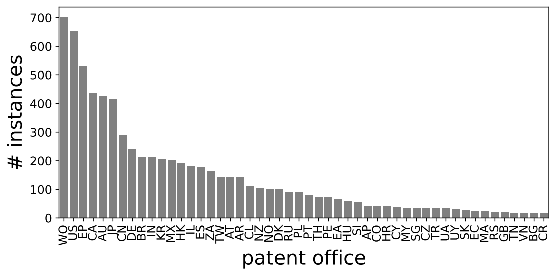


(c) Atazanavir

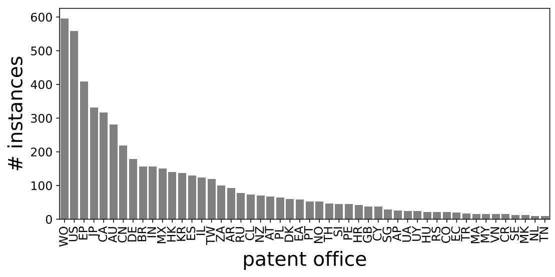
Figure 4.8: Instances per year for PLS datasets. The InjVal dataset is for a much longer time horizon (around 100 years) than the Rito and Atz datasets.



(a) Injection-Valve



(b) Ritonavir



(c) Atazanavir

Figure 4.9: Count of patent offices for the three PLS datasets.

Dataset	Documents	Unique Labels	Labels Per Instance
InjVal	9465	5068	6.42
Rito	781	3543	26.87
Atz	640	3171	31.18

Table 4.6: CPC/IPC statistics for the PLS datasets.

(Figure 4.8b, 4.8c) have a narrow timeline of roughly 20 years (16 and 22 years for Atz and Rito, respectively).

Patent Office / Original Language. To safeguard business interests, an organization may file an invention across multiple different patent offices around the globe (Carsten Fink and Zhou, 2016). The multiple patent filings are associated with a common patent family identifier. Figure 4.9 shows the distribution of published patents across different jurisdictions. It is interesting to note that most of the publications for the InjVal dataset (Figure 4.9a) are filed in Japan (JP) and Germany (DE), two primary hubs of industrial innovation. The WIPO datasets (Figure 4.9b, 4.9c) contain high numbers of worldwide filings (WO), with the United States (US) as the second most popular choice for filing a patent. This difference in jurisdiction indicates the document's language: most of the documents within the InjVal dataset are in non-English languages, which is not the case for the WIPO datasets. The majority of the patents (68%) in InjVal consist of machine-translated text.

CPC Labels. Table 4.6 shows the CPC statistics for the three PLS datasets. The patents in the WIPO datasets have a higher number of labels than those in the InjVal dataset. The InjVal dataset contains only one patent with a CPC count of more than 50, whereas Rito and Atz contain 13 and 18 such patents, respectively. In addition, the numbers of unique CPC labels within the WIPO datasets are relatively high compared to the InjVal dataset, given the relatively smaller sizes of the datasets.

4.5 Analyzing Association Between CPC/IPC Labels and PLS-Oriented Target Labels

Before selecting a document representation technique for a classification task, it is crucial to determine the dataset characteristics that might influence a classifier's performance for a given document representation technique. For PLS datasets, one such characteristic is the association between the CPC/IPC labels and the PLS-oriented target labels, which can influence the performance of document representation methods in the PLS classification task, depending on whether or not the document representation method uses CPC/IPC label information. Here, we hypothesize that classifiers exploiting CPC/IPC information will perform better in cases of higher association. To capture the association between a pair of CPC/IPC labels and a target label, we calculate the Pointwise Mutual Information (PMI)

(Church and Hanks, 1990) for each CPC/IPC-target pair. In this section, an analysis is performed on the three datasets as described in the following.

Comparing Association between CPC/IPC and PLS-oriented Labels. The PMI value is calculated between each CPC/IPC and PLS-oriented label pair and analyzed to determine the CPC/IPC and target label association for each of the three datasets. As the first analysis, the PMI values for pairs of top-25 CPC/IPC labels and PLS-oriented target labels are computed. The underlying assumption is that if a CPC/IPC label is essential for a target label, it is also essential for the dataset, as it plays a vital role in identifying the target label in the CPC classification task.

In addition, a domain expert can use the analysis in Figure 4.10 to understand the association between CPC/IPC and PLS-oriented labels. The label names and associated CPC/IPC label descriptions are more interpretable in the case of the InjVal dataset. Here, we consider a few examples of high-association pairs. The Piezoelectric Actuator Spring label shows a high association with F16F1/02[17] and F16F1/028[18], which have descriptions containing the text "Springs made of steel ...". Similarly, the Heat Pressure Pipe label shows a high association with F16L19/0283[19], which has a description including the text "... Pipe ends provided with collars or flanges ...".

Figure 4.10 shows the PMI values for target labels and top-25 CPC/IPC labels for all three datasets. In InjVal (Figure 4.10a), many CPC/IPC-target pairs have very high PMI values, considerably higher than in the other two datasets. For Rito (Figure 4.10b), fewer CPC/IPC-target pairs show high PMI values compared to Atz (Figure 4.10c).

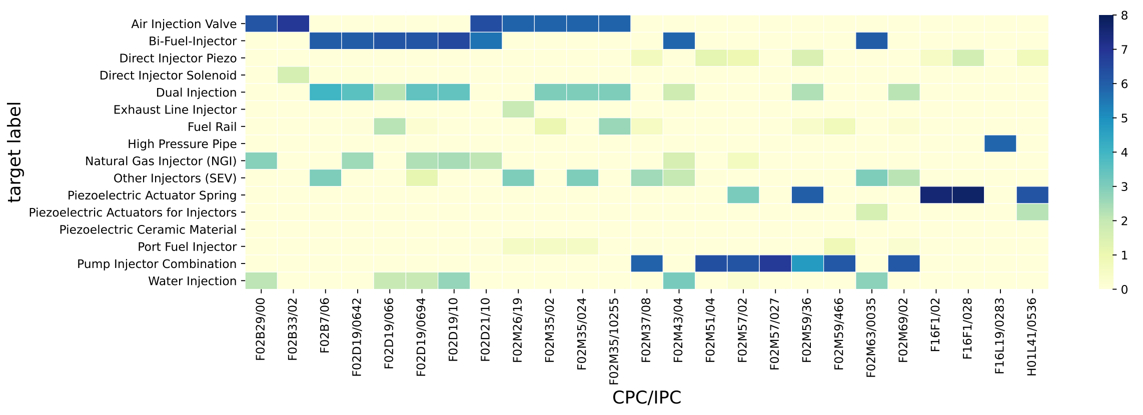
Variation of PMI Scores for Top-k. Apart from looking into the association between pairs of CPC/IPC and target labels, the mean PMI value is analyzed for the top-k CPC/IPC labels for three datasets and is shown in Figure 4.11. The InjVal dataset shows a much higher mean PMI score across the top-k CPC/IPC label counts than the WIPO datasets. Among the WIPO datasets, Rito's mean PMI score is higher than that of Atz.

Summary. The analysis above shows that the InjVal dataset exhibits a higher association between CPC/IPC and target labels than Rito and Atz. Of the two WIPO datasets, Rito shows a higher association than Atz. In Section 7.5, these characteristics will be used to better interpret results when evaluating and comparing different classification methods that use CPC/IPC document representation techniques.

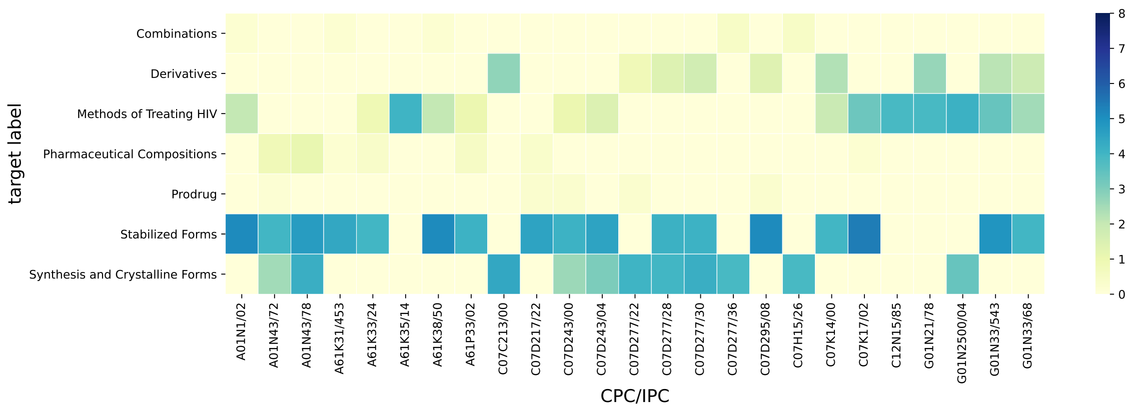
[17]https://www.uspto.gov/web/patents/classification/cpc/html/cpc-F16F.html#F16F1/02 [last accessed December 10, 2023]

[18]https://www.uspto.gov/web/patents/classification/cpc/html/cpc-F16F.html#F16F1/028 [last accessed December 10, 2023]

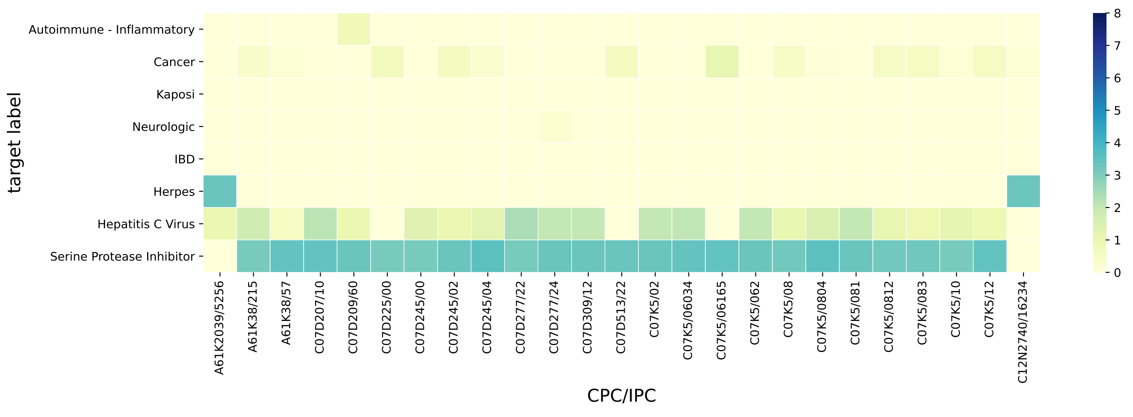
[19]https://www.uspto.gov/web/patents/classification/cpc/html/cpc-F16L.html#F16L19/0283 [last accessed December 10, 2023]



(a) InjVal



(b) Ritonavir



(c) Atazanavir

Figure 4.10: The association between the top-25 CPC/IPC labels and PLS-oriented categories is calculated using Pointwise Mutual Information (PMI).

4.6 Analyzing Duplicate Texts in Patents

Within a patent, information is distributed across multiple sections where a section text might provide an invention summary or a detailed specification, depending on the role associated with a section. For example, the abstract summarizes an invention, whereas the claims section provides a detailed specification. Since different sections convey information about the same invention, the text might be redundant or even duplicated across sections. Therefore, the performance of a classifier might not be improved by providing additional field text if the text contains redundant information.

As the same information can take multiple textual forms, identifying similar or near-duplicate text pairs is a challenging research problem (Bayardo et al., 2007; Feng and Deng, 2021). Some of the methods for identifying similar texts include: the shinglings-based method (Broder, 1997), the minhash algorithm (Broder et al., 1998), the locality-sensitive hashing (Gionis et al., 1999), and the Siamese network (Reimers and Gurevych, 2019). The reuse of text within and across documents is typical for patents. Therefore, we are interested in exact duplicates rather than the identification of near-duplicate text pairs when analyzing patent texts. We analyze patent texts from abstracts and sentences for duplicates and report our findings in the following.

Given a set of text chunks, we can determine duplicates by mapping each text chunk to a numerical hash value by using a standard hash function provided by the programming language and then counting the number of occurrences of a hash value. In this case, we use the built-in function from Python, which converts the input text to an integer value.[20] Since the release of version 3.4, Python has used the SipHash algorithm (Aumasson and Bernstein, 2012), which provides a better security guarantee against hash table collisions.[21] If a hash value occurs more than once, we can assume that the corresponding text is duplicated.

The titles, abstracts, and sentences are analyzed for duplicates. The analysis of duplicate titles and abstracts is performed using the USPTO-7M dataset to determine multiple occurrences of an abstract across patents. A sentence-level duplicate analysis is infeasible with the USPTO-7M dataset as it contains billions of sentences. Therefore, a sentence-level analysis is performed using the USPTO-70k dataset, which analyzes the occurrence of a sentence within and across documents. Below, we discuss the outcome of this analysis.

Duplicate Title and Abstract. By analyzing abstracts in the USPTO-70k dataset, we find that about 17% of the patents contain an abstract that is duplicated in at least one other patent, i.e., it appears as an abstract in at least one other patent. Figure 4.12 shows the duplicate count and the count of abstracts for each duplicate count value. As can be seen in Figure 4.12, approximately 380k abstracts occur in two patents, and one abstract is duplicated in 504 patents.

[20] https://docs.python.org/2/library/functions.html#hash [last accessed December 10, 2023]
[21] https://peps.python.org/pep-0456/ [last accessed December 10, 2023]

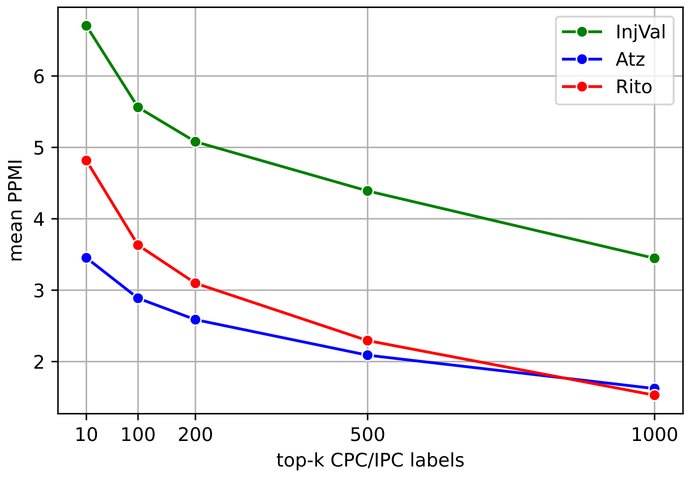


Figure 4.11: Mean Pointwise Mutual Information (PMI) values for top-k labels. The InjVal dataset has much higher PMI values than the WIPO datasets. Of the two WIPO datasets, Rito has higher PMI values than Atz.

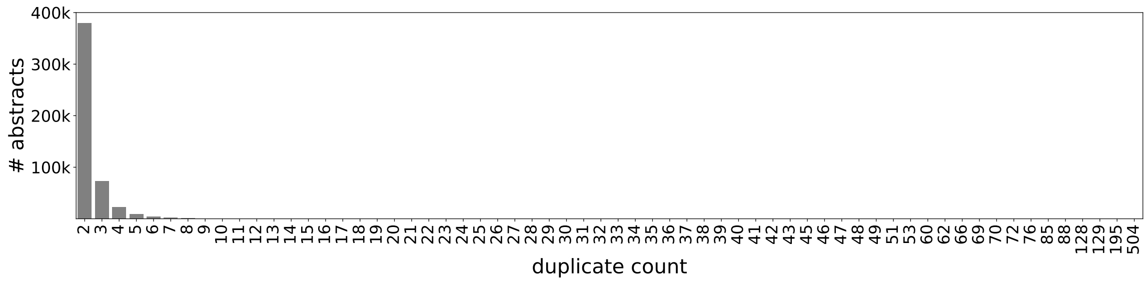


Figure 4.12: Number of abstracts corresponding to the duplicate count values for the USPTO-7M dataset.

By analyzing titles, we find that they are crafted with minimal word usage, e.g., “Tire”, “Bottle”, “Chair”, often with the intention of strengthening the claim over an invention. Table 4.7 shows the most frequent titles and their counts, demonstrating that some of the titles are used within thousands of patents. Through this analysis, we find that for 36% of patents, the title associated with a patent appears in at least one more document.

CPC Mismatch Between Documents with Same Abstracts. The analysis above reveals that abstracts are often reused across patents. Since a set of CPC labels represents the topic of a document, it might be interesting to check the number of cases in which patents with the same abstract are labeled with a different set of CPC labels—this is referred to as CPC mismatch. The analysis reveals a lesser chance of CPC mismatch between patents with abstracts having lower duplicate counts than those with abstracts having higher duplicate counts. For example, in 6% of the cases, patents with abstracts having a duplicate count of two show a CPC mismatch, whereas for a duplicate count of three, the proportion of such instances is 9%. Although these are small proportions, the occurrence of CPC mismatches

Title	Count
Semiconductor device	6955
Bottle	5307
Image Forming Apparatus	5119
Chair	4134
Display Device	4082
Container	3903
Electrical Connector	3070
Liquid Crystal Display Device	2755
Golf Club Head	2455
Display Screen or Portion thereof with Graphical User Interface	2427
Electronic Device	2399
Semiconductor Memory Device	2387
Mobile Phone	2348
Tire Tread	2339
Shoe Upper	2308
Tire	2069
Display Screen with Graphical User Interface	1970
Semiconductor Device and Method of Manufacturing the same	1915
Connector	1888
Shoe	1850

Table 4.7: Duplicate titles count for top-20 titles for USPTO-7M.

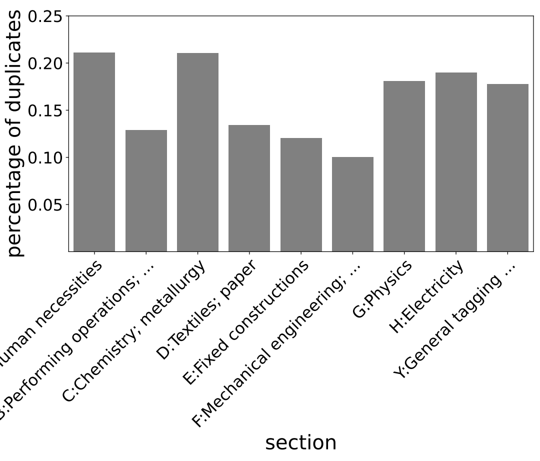


Figure 4.13: Percentage of patents with duplicate abstracts for different technical fields where a technical field corresponds to the first level labels of the CPC taxonomy. The statistics are computed for the USPTO-7M dataset.

indicates that abstracts are often written less carefully than other sections of patents and motivates us to look beyond titles and abstracts as input.

Topical Distribution of Duplicate Abstracts. To determine whether the tendency to duplicate abstracts is more prominent in particular technical fields or is the same irrespective of the technical field, we perform a topical analysis determining the proportion of duplicate abstracts within a domain. A topic-based analysis is performed, with the top-level CPC labels representing the nine technical categories. Figure 4.13 shows the percentage

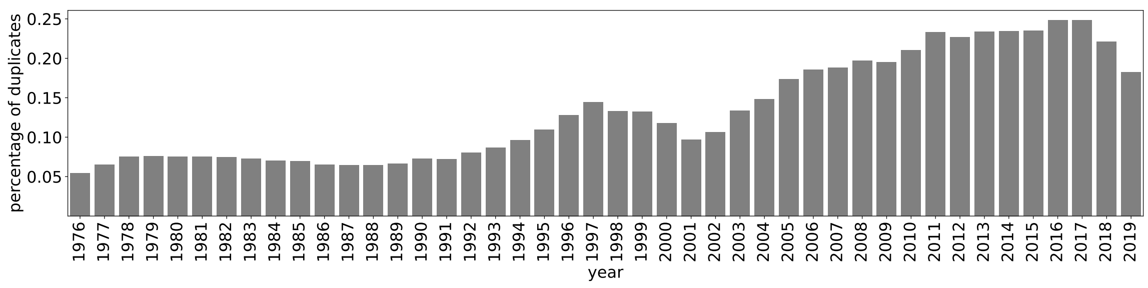


Figure 4.14: Temporal distribution shows the percentage of patents with a duplicate abstract out of the total number of patents published in a year for USPTO-7M.

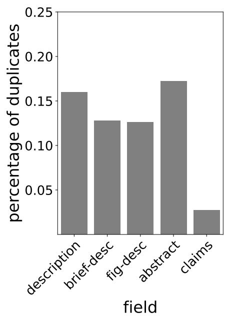


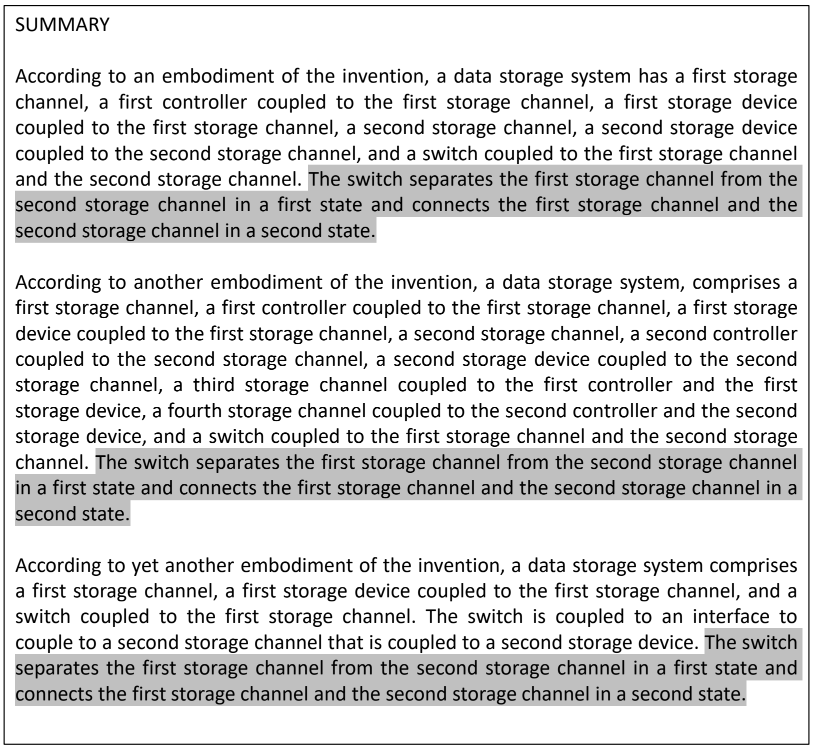
Figure 4.15: Percentage of duplicate sentences for each patent field in USPTO-70k.

of patents with duplicate abstracts for particular technical categories. The analysis reveals that in certain technical fields, inventors tend to reuse the abstract more often than they do in other fields. For example, the section labels "A" and "C", representing "Human Necessities" and "Chemistry; metallurgy", respectively, show a higher percentage of duplicates, whereas the label "F", representing the category of "mechanical engineering", has the lowest percentage of duplicate abstracts.

Temporal Distribution of Duplicate Abstracts. Figure 4.14 plots the percentage of patents with duplicate abstracts versus the total number of patents granted in a year. This temporal analysis of duplicate abstracts shows an interesting trend: the tendency to reuse abstracts has increased in recent years. In particular, since 2011, a high proportion of new patents (20% or more) have contained a duplicate abstract, whereas in the initial years (1976 - 91), only 7% or fewer patents had duplicate abstracts.

Analysis of Duplicate Sentences. In addition to duplicate abstracts, patents often contain sentences that are reused within or across documents. To understand the extent to which sentences are duplicated within and across patents, we analyzed 23M sentences within the USPTO-70k dataset.

A patent attorney drafts the patent application describing an invention, often using text that is similar or exactly the same as text in the same or other patents. The claims section is the only section that can be challenged with litigation, and therefore, the other sections


SUMMARY

According to an embodiment of the invention, a data storage system has a first storage channel, a first controller coupled to the first storage channel, a first storage device coupled to the first storage channel, a second storage channel, a second storage device coupled to the second storage channel, and a switch coupled to the first storage channel and the second storage channel. The switch separates the first storage channel from the second storage channel in a first state and connects the first storage channel and the second storage channel in a second state.

According to another embodiment of the invention, a data storage system, comprises a first storage channel, a first controller coupled to the first storage channel, a first storage device coupled to the first storage channel, a second storage channel, a second controller coupled to the second storage channel, a second storage device coupled to the second storage channel, a third storage channel coupled to the first controller and the first storage device, a fourth storage channel coupled to the second controller and the second storage device, and a switch coupled to the first storage channel and the second storage channel. The switch separates the first storage channel from the second storage channel in a first state and connects the first storage channel and the second storage channel in a second state.

According to yet another embodiment of the invention, a data storage system comprises a first storage channel, a first storage device coupled to the first storage channel, and a switch coupled to the first storage channel. The switch is coupled to an interface to couple to a second storage channel that is coupled to a second storage device. The switch separates the first storage channel from the second storage channel in a first state and connects the first storage channel and the second storage channel in a second state.

Figure 4.16: Example of a sentence appearing multiple times within US6983343B2. As shown in the figure, it is repeated three times within the summary (a subsection of the description). The same sentence also appears in the abstract section.

are written less carefully and contain relatively higher percentages of duplicate sentences. Figure 4.15 shows the proportion of duplicate sentences in each field. The percentage of duplicates is highest in the abstract field, followed by detail-desc, whereas the claims section contains the lowest proportion of duplicate sentences.

Further, we checked the proportion of duplicated sentences within and across documents. Analyzing 23M sentences, we found that 15% of sentences are duplicated in the corpus. In 36% of cases, a sentence is reused within the same document, and in 64% of instances, a duplicate sentence occurs across multiple documents. Figure 4.16 shows an instance where a sentence is used three times in consecutive paragraphs within the summary field. Furthermore, the sentence also appears in the abstract of the same patent.

In some instances, even though a duplicate sentence seems to be a technical system specification, it is repeated multiple times across patents, some of which belong to different applicants. For example, the sentence below is duplicated in as many as 700+ patents.

“ *In the latter scenario, the remote computer may be connected to the user’s computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the internet using an internet service provider).* ”

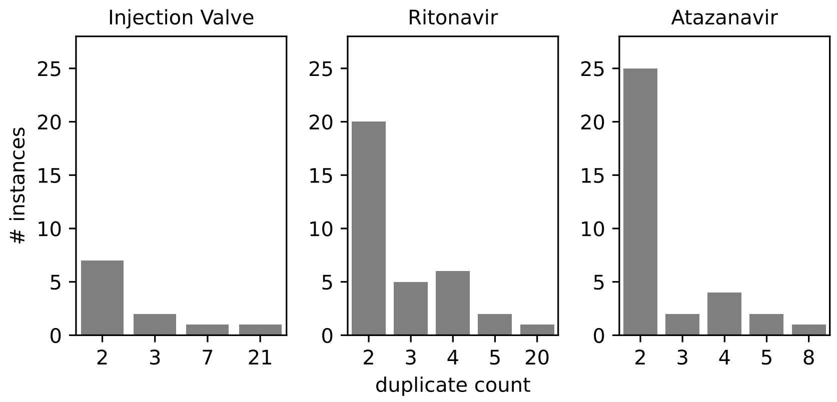


Figure 4.17: Duplicate abstract count for PLS datasets.

Analyzing Duplicate Abstracts in PLS Datasets. In the PLS datasets, 48, 109, and 90 instances in InjVal, Atz, and Rito do not have unique abstracts. As shown in Figure 4.17, the InjVal dataset contains seven abstracts that occur in at least two documents, whereas in the case of Rito and Atz, the number of such abstracts is 20 and 25, respectively. Some abstracts in InjVal and Rito occur up to 20 times (Figure 4.17).

We see two possible reasons for the higher number of duplicate abstracts in the Rito and Atz datasets compared to InjVal. First, by referring to Figure 4.10b and 4.10c, it can be concluded that the majority of patents in Rito and Atz are labeled with “A” and “C” category CPC/IPC labels, which have high duplicate abstract counts (Figure 4.13). By contrast, most instances in the InjVal dataset contain CPC/IPC labels belonging to the “F” category (Figure 4.10a), which has the lowest proportion of duplicates among all the Section labels (Figure 4.13). Secondly, the Rito and Atz datasets include patents relating to a very narrow scope concerning research and development around a single invention, which increases the potential for similar patents to be added with duplicate abstracts.

Summary. The analysis of duplicate texts reveals that the problem is prevalent across datasets and motivates us to use additional patent fields or even the full texts of patents for classification.

4.7 Conclusions

One of the major roadblocks for the development of patent classification models is the unavailability of appropriate datasets. Addressing that, we release USPTO-70k and three PLS datasets. Each instance in the USPTO-70k dataset contains the full text of the patent, and USPTO-70k is used to evaluate the CPC classification models in Chapters 5 and 6. The PLS classification models proposed in Chapter 7 are evaluated using three PLS datasets (see Section 7.5). Furthermore, we analyzed patent text to detect duplicates and found that sentences are often reused within and across patents. Additionally, the same abstract might appear in multiple patents. Our analysis of duplicate text indicates the redundant nature of patent texts and motivates the document representation methods proposed in Chapters 6 and 7. Finally, by analyzing the association between CPC/IPC labels and PLS-oriented categories, we find that the InjVal dataset has a higher association between them than the two WIPO datasets. This finding helps us interpret the results for the document representations that use CPC/IPC labels in Section 7.5.

Chapter 5

CPC Classification using Transformers

Although the CPC/IPC classification task is a hierarchical multi-label classification problem, most previous works using neural network architecture have proposed *flat classifiers* that predict leaf-level labels, usually at the third level of the taxonomy (Li et al., 2018a; Lee and Hsiang, 2019; Zaheer et al., 2020). On the other hand, TwistBytes (Benites, 2019), a hierarchical *local classifier per node* (LCN) approach that trains a non-neural classifier for each label in the taxonomy and predicts labels top-down performs better than the neural baselines when evaluated on the task of classifying the text of German 'blurbs' (Remus et al., 2019). A neural LCN-based approach using a transformer-based language model may perform better than a flat classifier on a hierarchical classification task. However, training multiple transformer-based classifiers in parallel is infeasible because of the high memory requirements. Addressing this issue, we propose a memory-efficient model architecture that trains multiple classification heads, one for each label in the taxonomy. The classification heads share a single transformer-based language model. Further, we use the hierarchical taxonomy structure in model architecture to improve prediction for labels at the lower levels of the taxonomy.

In Section 5.1, we highlight the challenges associated with the application of a neural LCN approach when using a transformer-based language model (Devlin et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020) for text representation. The limitations of the current state-of-the-art patent classification approaches and other hierarchical methods are discussed in Section 5.2. The architectural details for TMM and THMM are described in Section 5.3. The proposed techniques are compared against neural and non-neural baselines (Section 5.5) using the experimental setup defined in Section 5.4.

5.1 Motivation and Contributions

Predefined patent taxonomies, for example, the Cooperative Patent Classification (CPC) scheme (see Section 2.1.3), arrange labels in a hierarchical taxonomy structure, such that the labels representing coarse-grained concepts are at higher levels of the hierarchy, whereas those representing fine-grained concepts are at the lower levels. Although CPC/IPC classification is a hierarchical multi-label classification task, most previously-developed models have ignored the hierarchical structure when defining the model architecture and predicted labels for only a single level of the class hierarchy, mainly at the third level of the taxonomy (Li et al., 2018a; Lee and Hsiang, 2019). Such classification techniques are referred to as *flat classifiers* by Silla and Freitas (2011). We hypothesize that a hierarchical classifier that learns concepts at each level of the hierarchy will perform better than a flat classifier when evaluated on a hierarchical classification task. Even if misclassification occurs at a lower level when using this kind of classifier, correct prediction at a higher level may improve prediction accuracy. Furthermore, the use of hierarchical links in the model architecture can improve prediction at lower levels (Benites, 2019; Wehrmann et al., 2018).

Silla and Freitas (2011) categorized hierarchical classifiers into three main groups: *Local Classifiers per Level* (LCL), *Local Classifiers per Node* (LCN), and *global* classifiers. An LCL approach trains a classifier for each level of the class hierarchy that predicts the labels within a level. In contrast, an LCN approach trains a classifier for each label in the taxonomy. A global approach predicts all the labels in the taxonomy across levels.

Benites (2019) propose a non-neural LCN-based approach for training a classifier for each label in the taxonomy. It takes a TF-IDF feature vector as input and has been shown to outperform a flat neural approach (Hepburn, 2018) based on the ULMFiT contextual language model (Howard and Ruder, 2018) when evaluated on the task of classifying the text of German 'blurbs' (Remus et al., 2019). Further, as a state-of-the-art patent classification model, a flat classifier (Lee and Hsiang, 2019) representing text with a transformer-based language model (Devlin et al., 2019) exhibited superior performance when compared to a CNN-based classifier (Li et al., 2018a) that used non-contextualized word embeddings (Mikolov et al., 2013b). Inspired by the success of Benites (2019), and Lee and Hsiang (2019), our objective is to create an LCN-based neural classifier that incorporates a transformer-based language model for text representation. Such a neural model architecture would include multiple classifiers, one for each label in the taxonomy, and use a transformer-based language model for text representation. However, given the high memory requirements for a transformer-based model, the parallel training of multiple label-specific classifiers is infeasible. Thus, we see a need for an innovative memory-efficient approach.

Combining the benefits of the approaches taken by Benites (2019), and Lee and Hsiang (2019), in this chapter, we propose a novel memory efficient model architecture, namely the *Transformer-based Multi-task Model* (TMM). TMM trains a classification head for each label in the taxonomy, and all classification heads share a single transformer-based language model for text representation. Further, motivated by the hypothesis that learning from a

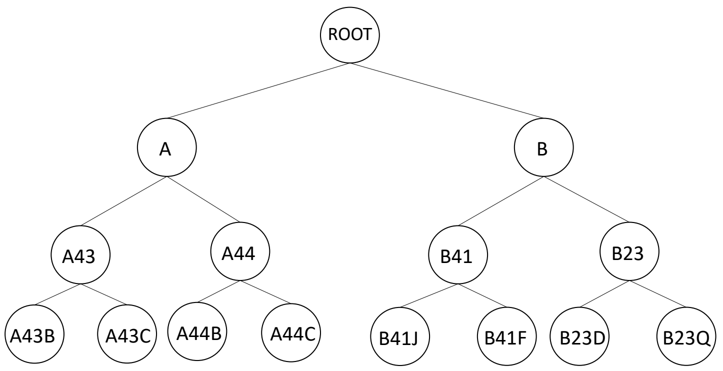


Figure 5.1: Example of CPC taxonomy showing the hierarchical structure up to the subclass labels, i.e., the third level of the taxonomy.

parent task might benefit a child task, we transfer the hidden state of the corresponding parent head to those of its children. This hierarchical variant of TMM is referred to as the *Transformer-based Hierarchical Multi-task Model* (THMM).

Our approach is the first to combine powerful transformer-based language models with an intrinsically hierarchical algorithm for patent classification. The evaluations are performed with the USPTO-70k (see Section 4.3) and WIPO-alpha (Section 5.4.1) datasets. Evaluation of the proposed approach shows that it performs better than both neural (Li et al., 2018a; Lee and Hsiang, 2019; Huang et al., 2019) and non-neural baselines (Benites, 2019). Comparing THMM and TMM, we find that by using hierarchical links, THMM makes better predictions at lower levels, which contain many infrequent labels.

Contributions. We make two main contributions in this chapter, which can be summarized as follows:

- We propose a **novel neural LCN approach** that efficiently trains multiple classification heads using a shared transformer-based language model for text representation, referred to as the Transformer-based Multi-task Model (TMM) (see Section 5.3). The proposed architecture combines the advantages of powerful document embeddings generated by a pre-trained language model with the gains that can be achieved by localizing decisions.

- In addition, we propose a **hierarchical variant** that transfers the learned representation from the parent head to the heads corresponding to its children (see Section 5.3). Further, an in-depth analysis demonstrates that the proposed model strongly outperforms previously developed models, achieving much **higher classification accuracy** at the **lower levels of the hierarchy** and **for infrequent labels**.

5.2 Related Work

Prior works on patent datasets are discussed in Section 4.2, whereas those on patent representation methods are reviewed in Sections 6.2 and 7.2. Here, we discuss the key classification model architecture applied for patent classification. Further, we look into some recent works on hierarchical text classification.

Gomez and Moens (2014) provide a detailed survey of various patent classification techniques. Prior non-neural works have used a large variety of classification models including Winnow (D'hondt et al., 2013, 2014), Naive Bayes (Lim and Kwon, 2016), Support Vector Machines (Chen and Chang, 2012), k-Nearest Neighbors (Gomez, 2019), and Logistic Regression (Gomez, 2019). The neural network classifiers primarily use feedforward neural networks that take a patent representation as input and generate prediction scores corresponding to each label in the taxonomy (Li et al., 2018a; Lee and Hsiang, 2019; Zaheer et al., 2020; Choi et al., 2022). For example, Lee and Hsiang (2019) propose a technique that takes the truncated patent text as input, generate a sequence representation using BERT (Devlin et al., 2019), and pass it through feedforward neural networks, fine-tuning BERT model weights in the training process.

In the ALTA shared task on patent classification (Mollá and Seneviratne, 2018), an approach training separate SVM classifiers per node using simple n-gram and POS-tag based features (Benites et al., 2018) performed comparably to a flat neural approach (Hepburn, 2018) based on the ULMFiT contextual language model (Howard and Ruder, 2018). The work of Li et al. (2018a), based on Kim (2014) and optimized by Abdelgawad et al. (2019), proposes a convolutional neural network (CNN) based approach which predicts IPC codes at the subclass level and uses the non-contextual word2vec embeddings (Mikolov et al., 2013b). Further, neural work on patent classification employs graph-convolutional networks using word embeddings inferred from a word-document co-occurrence graph (Tang et al., 2020).

The CPC/IPC classification models proposed in this chapter are compared to the hierarchical neural classifier HARNN (Huang et al., 2019), a global neural classification approach predicting labels across the hierarchy. HARNN is one of the first neural hierarchical approaches that was evaluated on the CPC classification dataset. HARNN generates a document embedding by aggregating token embeddings initialized with word2vec using a BiLSTM and feeds it through a hierarchical attention-based memory unit that learns different attention weights per category. The final prediction combines hidden *local* and *global* information (see Section 2.6.3). Hierarchical patent classification has also been addressed as a sequence generation problem using an attention-based neural model (Risch et al., 2020b).

Our approach (Pujari et al., 2021a) differs from prior work for the following reasons. First, instead of predicting labels at a single hierarchical level (Abdelgawad et al., 2019; Benites et al., 2018; Lee and Hsiang, 2019; Li et al., 2018a), we model predictions across the label taxonomy. Second, we use the hierarchical taxonomy structure and connect heads corresponding to those of their children, unlike the flat classification model architectures

(Lee and Hsiang, 2019; Li et al., 2018a). Third, we used contextualized embeddings for text representation instead of using non-contextualized embeddings (Li et al., 2018a; Huang et al., 2019; Risch et al., 2020b).

Below, we describe some key approaches that train level-specific neural classifiers and apply them for tasks other than patents. Similar to our work, Peng et al. (2018) propose a CNN-based model in which the hierarchy of labels is leveraged by regularizing the deep architecture with dependencies among labels. Kowsari et al. (2017) and Wehrmann et al. (2018) address neural hierarchical text classification by training level-wise classifiers and chaining predictions from the top down in a single hierarchical model. In contrast, Banerjee et al. (2019) and Shimura et al. (2018) train separate classifiers for each level and use transfer learning, initializing the lower level classifier using the parameters learned at the higher level. Banerjee et al. (2019) train a binary classifier for a parent label and use its parameters to initialize a classifier for the child labels. Shimura et al. (2018) fine-tune a CNN-based model to predict labels at a certain level, and the parameters corresponding to the CNN filters are transferred to initialize layers at the lower level. Xu et al. (2021) enhance Wehrmann et al. (2018) by using horizontal and vertical correlation between labels, i.e., correlations between labels both within and across levels. Contrary to the level-based approaches described above, we propose a model architecture similar to the local classifier approach in which multiple classification heads are trained, one corresponding to each label in the taxonomy. The classification heads share a common language model for text representation. Further, we leverage the hierarchical taxonomy links by transferring representation from the classification head for the parent to the heads corresponding to its children.

Above, we discussed the key methods with high relevance to our proposed approach. Next, we look into some of the recent hierarchical classification approaches, which show a huge variation in their methodologies. Meng et al. (2019) and Shen et al. (2021) suggest weakly-supervised hierarchical classification approaches. Given a few user-provided seeds, Meng et al. (2019) propose a system that generates pseudo-documents and uses them to bootstrap a neural hierarchical classifier which includes an LSTM-based language model. Shen et al. (2021) propose an approach in which a label is assigned based on the output of a pre-trained entailment model for document text, with label description as input. The process is performed from the top down, checking entailment only for the children of labels that were found to be important in the level above.

Some works on hierarchical classification use label co-occurrence statistics (Zhou et al., 2020; Chatterjee et al., 2021). Zhou et al. (2020) compute the priors for labels conditioned on their parents and use them as edge weights of hierarchical taxonomy links to encode the label hierarchy using Bidirectional Tree LSTMs (Li et al., 2018b) and GCNs (Kipf and Welling, 2017). Chatterjee et al. (2021) propose a hierarchical model for situations in which the label hierarchy is unknown. They project labels in a hyperbolic space, as this can better capture hierarchical relations and use a loss that minimizes the distance between co-occurring labels.

Few methods encode the hierarchical taxonomy using GCNs and leverage it to enrich text representation during training (Chen et al., 2021a; Wang et al., 2022a). Chen et al. (2021a) encode text using CNNs and encode labels using GCNs. Further, they align text and label embeddings using a loss that minimizes the distance between text embeddings and positive labels and penalizes short distances to negative labels. Wang et al. (2022a) enrich the text encoder with hierarchical taxonomy information during training by employing contrastive learning.

Rivas Rojas et al. (2020) address hierarchical classification as a sequence-to-sequence prediction task, predicting a label at a level in each step. Similarly, Huang et al. (2022) model hierarchical classification as a generative task in which labels are predicted one level at a time, and label dependency is used across different levels. For this, they use path-augmented attention by focusing on labels that are within the path of the label predicted in the previous step. Jiang et al. (2022) use BERT to learn label embeddings through the masked label prediction task, a method similar to the masked token prediction task (Devlin et al., 2019). The attention mask restricts attention to the parent and child labels in the hierarchy.

Inspired by adversarial learning techniques (Makhzani et al., 2016), Deng et al. (2021) use a discriminator to maximize the mutual information between a label and the actual text and minimize it for text corresponding to a randomly picked sample of the mini-batch. Wang et al. (2021) propose a concept routing approach similar to CapsNet (Sabour et al., 2017), in which the top keywords from the document are picked as concepts.

Few recent approaches apply soft prompting to incorporate label information into document representations (Chen et al., 2022; Wang et al., 2022b). Chen et al. (2022) propose a method that uses a label-specific soft prefix prompt vector, which incorporates the hierarchical structure of the taxonomy. Wang et al. (2022b) introduce an input template with slots for soft prompt vectors corresponding to each hierarchical level, in which a level-specific vector is aggregated using the graph attention network.

5.3 Model Architecture

Here, we provide an overview of the TMM model architecture (Section 5.3.1), followed by specific details on document representation (Section 5.3.2) and the classification model (Section 5.3.3).

5.3.1 Overview

A classification pipeline consists of two main parts: a document representation method and a classification model. Following previous neural and non-neural classifiers (Li et al., 2018a; Lee and Hsiang, 2019), we use patent title and abstract as inputs to the classifier. As a first step, a distributed representation is created for the textual input using a pre-trained transformer-based language model and provided as input to the classification model. The

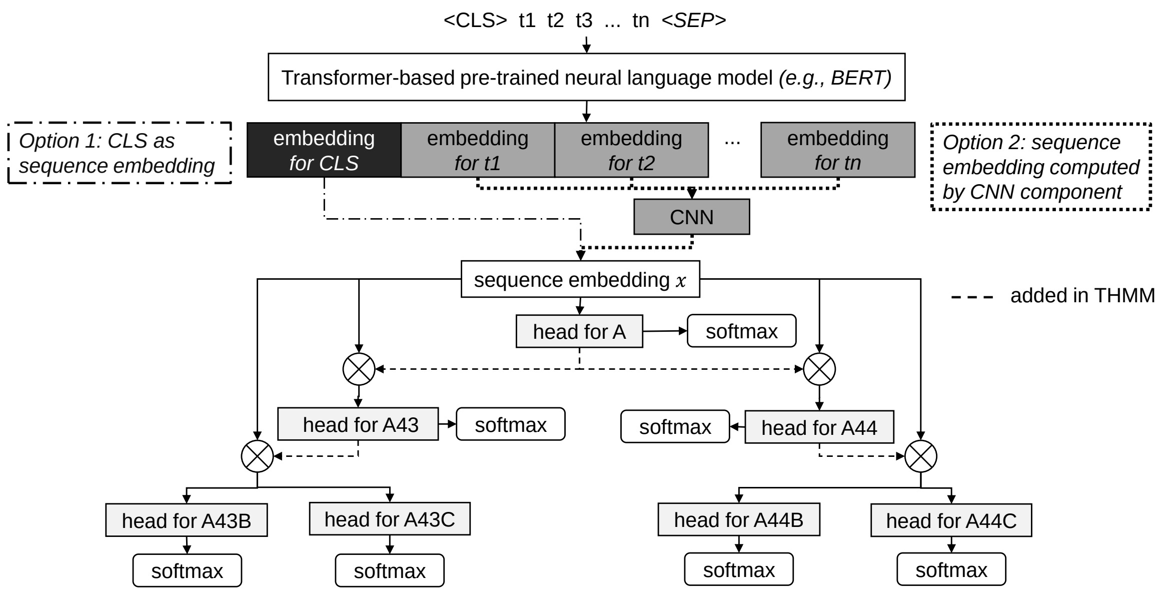


Figure 5.2: Model architecture for the Transformer-based Hierarchical Multi-task Model, corresponding to the left subtree of the sample taxonomy shown in Figure 5.1 with root at node "A".

classification model trains a binary classifier for each label in the taxonomy, predicting whether an instance (i.e., a patent) belongs to the respective category or not. The ensemble of classifiers is trained in a multi-task setup and uses a single underlying SciBERT neural language model (Beltagy et al., 2019) to create document representations. SciBERT has been trained on a corpus of scientific publications and is thus closer to the patent domain than BERT (Devlin et al., 2019), which was trained on English-language Wikipedia and BookCorpus (Zhu et al., 2015) text.

In the terminology of multi-task learning, each of these classification heads addresses one *task*. Hence, each label-specific binary classifier constitutes a classification head in the multi-task-based neural network architecture. In other words, the *Transformer-based Multi-task Model (TMM)* consists of a single transformer model with n heads, where n corresponds to the number of labels in the hierarchy. Hard parameter sharing is used for the transformer model (and for optional CNN layers) (Ruder, 2017). In addition, each classification head has its own set of parameters.

We hypothesize that the classification head at the higher level captures coarse information on a document's topic, whereas the classification head at the lower level captures fine-grained information. The lower-level tasks may benefit from receiving additional semantic information from a parent classification head. To achieve this, the TMM architecture is extended to link the network components corresponding to parent and child nodes. The extended TMM version is called the *Transformer-based Hierarchical Multi-task Model (THMM)*.

5.3.2 Document Representation

Similar to prior works on neural patent classification (Li et al., 2018a; Huang et al., 2019; Lee and Hsiang, 2019), the title and $\mathrm{abstract}$ fields of a data object based on the patent document model (see Section 3.2.2) are provided as input to a patent classifier. The experiments are performed with two types of document representation. In the first case, the embedding for the [CLS] token from the last transformer layer of SciBERT is used as a document representation (see Section 2.4.3). However, the [CLS] token was designed for the next sentence prediction task, and it is unclear how effective its embedding is when representing long sequences, as in our case. Hence, the second option is to consider the entire sequence. In this case, an embedding is generated for each token in the sequence. Next, the token embeddings are provided as input to a CNN layer, which computes a position-invariant feature vector (right/dotted path in Figure 5.2). Further details on text representation methods are provided in Section 2.4.2.

As discussed in Section 3.2, the classification pipeline takes an instance of the domain-specific document model, $\mathrm{PatentDM}$ in this case, and maps it to an instance of the domain-independent document model, which is finally mapped to a instance of the data model that is specific to the document representation method. The document representation methods described here take an instance of $\mathrm{DocRepDM\_Seq}$ as input, which has text and $\mathrm{max\_length}$ as attributes (see Section 3.2.4). For the experiments performed in this chapter, the text corresponds to the concatenated text of title and $\mathrm{abstract}$ whereas the maximum sequence length value is specified as $\mathrm{max\_length}$.

5.3.3 Classification Models

We now provide a comprehensive overview of the architectural details of the classification models. An independent head is created for each label that takes document representation $\boldsymbol{x}$ as input, which corresponds to the [CLS] token embedding or the CNN's output. Each head consists of two dense layers with ReLU activation followed by a two-dimensional dense output layer that produces logits. The classification is performed by applying a softmax operation to the output logits.

Working on the assumption that a representation learned in the parent head might improve the performance of the child head, we propose a hierarchical variant of TMM that transfers the intermediate representation from the parent head to the child head and is referred to as the *Transformer-based Hierarchical Multi-task Model (THMM).* As in TMM, each classification head computes the logits for the binary decision using two fully connected dense layers. However, in THMM, the first hidden layer of the classification head for c_i also takes into account the logits $\boldsymbol{h}^3_{c_j}$ taken from the final (third) dense layer of the head corresponding to c_i's parent c_j. It computes a hidden representation $\boldsymbol{h}^1_{c_i}$ by performing a linear transformation on the concatenation (;) of the sequence embedding $\boldsymbol{x}$ and $\boldsymbol{h}^3_{c_j}$. If c_i does not have a parent in the taxonomy, the input to its classification head is $\boldsymbol{x}$. Further, $\boldsymbol{b}^1_{c_i}$

and $\boldsymbol{b}^2_{c_i}$ are the basis vectors corresponding to the hidden states $\boldsymbol{h}^1_{c_i}$ and $\boldsymbol{h}^2_{c_i}$, respectively, whereas $\boldsymbol{b}^3_{c_i}$ is the basis vector corresponding to the output logit vector $\boldsymbol{h}^3_{c_i}$.

$$\boldsymbol{h}^1_{c_i} := \begin{cases} \phi(\boldsymbol{W}^1_{c_i}(\boldsymbol{h}^3_{c_j};\boldsymbol{x}) + \boldsymbol{b}^1_{c_i}) & \text{if there is a } c_j \text{ with } parent(c_i, c_j) = true \\ \phi(\boldsymbol{W}^1_{c_i}\boldsymbol{x} + \boldsymbol{b}^1_{c_i}) & \text{if } parent(c_i, ROOT) \end{cases} \tag{5.1}$$

$$\boldsymbol{h}^2_{c_i} := \phi(\boldsymbol{W}^2_{c_i}(\boldsymbol{h}^1_{c_i}) + \boldsymbol{b}^2_{c_i}) \qquad \boldsymbol{h}^3_{c_i} := \boldsymbol{W}^3_{c_i}(\boldsymbol{h}^2_{c_i}) + \boldsymbol{b}^3_{c_i} \tag{5.2}$$

As in TMM, the hidden representation $\boldsymbol{h}^1_{c_i}$ is passed through two further dense layers and mapped to a two-dimensional logit vector $\boldsymbol{h}^3_{c_i}$. This serves as input to a softmax layer that predicts whether label c_i applies to the instance. We use binary cross-entropy loss to train our models and weigh all “tasks” equally.

5.4 Experimental Setup

The proposed models and baselines are evaluated on a CPC and an IPC dataset, as described in Section 5.4.1. This is followed by a description of evaluation metrics (Section 5.4.2), the specification of hyperparameters for the proposed model (Section 5.4.3), and baselines (Section 5.4.4).

5.4.1 Dataset

The models are evaluated on the USPTO-70k and WIPO-alpha datasets. The USPTO-70k dataset (see Section 4.3) contains about 70,000 patents. Each patent instance contains the full text of the patent and is labeled with labels taken from the third level of the CPC taxonomy (see Figure 2.3, page 15), i.e., subclass labels. The WIPO-alpha dataset contains about 46,000 training instances and 29,000 test instances (Fall et al., 2003). The patent documents were published between 1998 and 2002, with test instances sampled randomly. There are 602 labels in the training set and 576 labels in the test set at the subclass level. As there is no pre-existing split, we sample the validation set from the training set by selecting 20% of the data points at random. There are 22 labels with instances in the test set but none in the training set at the subclass level. The IPC code in the dataset is defined according to the seventh edition of the IPC, which labels each patent with a main IPC code and a set of secondary IPC codes. Unlike prior work by Abdelgawad et al. (2019), which considers only the main IPC code and benchmarks models in a single-label flat classification setting, we consider all IPC codes in a hierarchical multi-label classification setting.

5.4.2 Evaluation Metrics

Unlike previous works, which have predominantly used micro-average precision, recall, and F1 scores (Li et al., 2018a; Huang et al., 2019; Lee and Hsiang, 2019), we compute macro scores to better evaluate classifiers' performance on less-frequent labels. To evaluate the capability of a classifier to make a partially correct classification, we use the hierarchical evaluation metrics defined in Section 2.7. In the case of hierarchical evaluation metrics, the true label set consists of labels across the hierarchy. To evaluate a partially correct prediction, the ancestors of the predicted labels are also added to the predicted label set before calculating the evaluation score. We calculate micro- and macro-average scores for the precision, recall, and F1 metrics. The micro-average score captures the performance of a classifier across all test instances, whereas the macro-average score is an average of evaluation scores across labels. Thus, macro-average scores capture the ability of a classifier to perform well in a class-imbalance scenario.

5.4.3 Hyperparameters

The models are implemented in Python using TensorFlow 2.0[1] and Keras (Chollet et al., 2015). Further, the HuggingFace Transformers library (Wolf et al., 2019) is used to integrate SciBERT (Beltagy et al., 2019) into the model architecture. For efficiency reasons, the word-piece tokenized input sequences generated by a text concatenation of title and $\mathrm{abstract}$ are truncated to a maximum length of 256 tokens. As illustrated in Figure 4.5 on page 61, this covers the complete input text for almost all instances in the USPTO (and also for WIPO-alpha, not shown).

Due to high computation cost, the hyperparameters are fine-tuned on 5k randomly selected instances. The following hyperparameters achieve the best performance across two benchmark datasets for the proposed TMM and THMM models. All dense layers have a hidden size of 256 and use ReLU activations. Model training is performed with a learning rate of 10^{-5}, a dropout rate of 0.25 across layers, and a batch size of 64. The CNN variant largely follows the architecture used in Li et al. (2018a): a word-piece token embedding is generated by summing up the weights for the last four SciBERT layers, which are concatenated to generate an input tensor. Next, the CNN layer applies a convolution operation over the input tensor, with varying kernel sizes of $\{2, 3, 4, 5\}$, each capturing a location-invariant n-gram feature. In contrast to Li et al. (2018a), an extra kernel of size two is added to capture bigrams.

5.4.4 Baselines

We compare our models to a wide range of non-neural and neural classifiers, the details of which are provided below.

Non-neural. The **TwistBytes** system (Benites, 2019) constitutes a competitive *non-neural* baseline (see Section 2.6.3). The system is implemented using scikit-learn[2] and learns one

[1]https://www.tensorflow.org [last accessed December 10, 2023]

[2]https://github.com/globality-corp/sklearn-hierarchical-classification [last accessed December 10, 2023]

support vector classifier (see Section 2.6.1) per node. During prediction, the model only tests for the presence of labels if the respective parent's score is positive. Finally, the set of predicted labels is filtered using a threshold of -0.25, based on Benites (2019). Further details on the TwistBytes algorithm are provided in Section 2.6.3.

Neural. The performance of the proposed models is compared with three neural baselines, the details of which are provided in Section 2.6. The hyperparameters for each of these three approaches are specified below.

HARNN. The hyperparameter setting for HARNN is as proposed by Huang et al. (2019): each document uses a 100-dimensional word2vec model trained on the training and validation data, using 256 and 512 as the hidden sizes in the BiLSTM and for each fully connected layer, respectively. Local and global information are combined using a regulation parameter α with a value of 0.5. To establish a fair comparison with the other models, the prediction threshold is tuned for macro-performance by setting it to 0.15 for both datasets. **HARNN-orig** (Huang et al., 2019) uses a prediction threshold of 0.5. HARNN is described in detail in Section 2.6.3.

flat-*. In addition, we provide results for simplified versions of our model, which only predict labels for the leaf level and infer ancestors during post-processing. First, **flat-CNN** corresponds to DeepPatent (Li et al., 2018a), which uses a CNN with kernels of sizes {3, 4, 5} and 512 filters on top of SciBERT. The outputs of all CNN layers are flattened and concatenated, resulting in a 1,536-dimensional document embedding. The details of CNNs are provided in Section 2.4.2. Second, **flat-CLS** is based on PatentBERT (Lee and Hsiang, 2019), using SciBERT's 768-dimensional [CLS] embedding directly as document embedding. The feature vectors of **flat-CNN** and **flat-CLS** are subsequently fed into a multi-layer perceptron (see Section 2.6.2) with two dense layers, applying sigmoid activation to each logit. For both models, the dense layers have a size of 512, the learning rate is set to 10^{-5}, the dropout rate is 0.25, and the batch size is 64.

Runtime. Training the TwistBytes model takes 1 to 1.5 hours, whereas HARNN model training takes 10 to 12 hours. Training the transformer-based models takes 300 hours, much longer than either the TwistBytes or HARNN models. Each of the models is trained on a single Nvidia Tesla V100 GPU. The early stopping is employed if the macro-F1 score on the validation set does not improve for five epochs.

5.5 Results

The performance of the proposed models is compared against the baselines, and the results are shown below. Similar tendencies can be observed when analyzing the results for the USPTO-70k and WIPO-alpha datasets. The proposed model performs better than the baselines, especially at the lower levels of the hierarchy and for less frequent labels.

	macro-avg.			micro-avg.		
Model	**P**	**R**	**F1**	**P**	**R**	**F1**
TwistBytes (Benites, 2019)	42.3	20.3	25.7	65.1	53.4	58.7
HARNN-orig (Huang et al., 2019)	35.5	12.6	17.0	**78.1**	48.1	59.5
HARNN (Huang et al., 2019)	29.2	28.1	26.7	51.9	67.9	58.8
flat-CNN (Li et al., 2018a)	**48.6**	27.2	33.0	71.8	55.2	62.4
TMM-CNN	41.2	36.0	36.6	63.9	**63.6**	63.7
THMM-CNN	41.2	36.4	36.9	64.9	63.4	64.1
flat-CLS (Lee and Hsiang, 2019)	48.1	25.6	31.6	74.0	54.6	62.8
TMM-CLS	48.5	31.3	36.2	70.9	61.1	**65.6**
THMM-CLS	42.6	**36.7**	**37.7**	66.6	63.3	64.9

Table 5.1: Classification results on USPTO test set comparing the performance of the proposed models (TMM/THMM) against the flat neural (flat-CLS/flat-CNN), hierarchical neural (HARNN), and hierarchical non-neural (TwistBytes) baselines.

5.5.1 Classification Performance

Tables 5.1 and 5.2 show the results obtained for the USPTO and WIPO-alpha datasets, respectively. The following conclusions can be drawn from these results. First, neural models generally perform better than the non-neural TwistBytes system, with SciBERT-based models outperforming HARNN. The proposed models achieve much higher recall while maintaining high precision. When tuning HARNN for F1, as in the original work (Huang et al., 2019), high micro-P can be achieved at the cost of lower recall, especially in the macro evaluation.[3] This implies that the original model focuses on the easy cases of highly frequent labels. Tuning HARNN for macro-scores changes the precision-recall trade-off in the micro-setting and improves macro-F1, but the model still does not approach the performance of transformer-based models.

With the exception of the macro-P of flat-CNN on USPTO, the CLS-based models outperform their CNN-based counterparts. However, the CLS-based models achieve the best results in terms of micro- and macro-F1 on both datasets. These observations indicate that there is no need to aggregate additional information from the sequence using a CNN layer. In most cases, adding hierarchical links between classification heads in TMM increases recall at the expense of precision. When comparing THMM-CLS with TMM-CLS on both datasets, we found that the former does better in terms of macro-F1, while the latter has slightly higher micro-F1; this indicates that adding the links helps, especially for less frequent labels.

Finally, the flat strategy leads to good precision but is not competitive in terms of recall, demonstrating that such models struggle to activate all relevant classifications to the required extent. Hence, our experiments confirm that when optimizing for a good trade-off between micro- and macro-average performance, hierarchical multi-label classification for patents is best approached using a fully hierarchical model.

[3]We double-checked the surprisingly low macro-scores of HARNN-orig and decided to present results for a HARNN model tuned for macro-performance as well.

	macro-avg.			micro-avg.		
Model	**P**	**R**	**F1**	**P**	**R**	**F1**
TwistBytes (Benites, 2019)	45.6	26.4	30.8	62.6	57.0	59.7
HARNN-orig (Huang et al., 2019)	08.9	02.1	02.7	**75.7**	24.8	37.3
HARNN (Huang et al., 2019)	20.6	26.9	20.6	37.3	65.2	47.4
flat-CNN (Li et al., 2018a)	46.6	34.8	38.2	70.7	57.8	63.6
TMM-CNN	40.8	40.0	38.9	63.6	68.4	65.9
THMM-CNN	37.7	41.3	38.0	62.0	68.6	65.1
flat-CLS (Lee and Hsiang, 2019)	**50.3**	32.8	37.7	73.7	59.8	66.0
TMM-CLS	46.2	37.6	39.9	68.2	67.9	**68.0**
THMM-CLS	40.9	**42.4**	**40.5**	65.1	**69.8**	67.4

Table 5.2: Classification results for the WIPO-alpha test set, comparing the performance of the proposed models (TMM/THMM) against flat neural (flat-CLS/flat-CNN), hierarchical neural (HARNN), and hierarchical non-neural (TwistBytes) baselines.

Figure 5.3 shows an increase in macro-F1 for TMM and THMM compared to the baselines, which is primarily a result of higher recall (not shown). Adding hierarchical links (i.e., THMM rather than TMM) results in better predictions, mainly at level 3. Hence, the overall increase in F1 is a result of improved classification at the lower levels, and the classification of finer-grained labels benefits from the passing on of hierarchical information.

5.5.2 Coverage

The number of labels at the subclass level varies strongly across instances, from a single category to 20 or more, with a tendency for more recent patents to have more labels. Therefore, it is challenging to output the right number of categories per instance in this task (Fall et al., 2003). To evaluate the ability of classifiers to predict an adequate number of labels at a certain level, we analyze the number of labels predicted by each classifier at each level. Figure 5.3 breaks down the average number of labels predicted by the level of the hierarchy for USPTO (WIPO-alpha shows similar tendencies). At the top level of the hierarchy, all models predict roughly the same number of labels as in the true label set. However, at levels 2 and 3, TwistBytes and the flat models predict markedly fewer labels. The TMM and THMM models alleviate this effect. While HARNN-orig strongly under-predicts the number of labels, the version of HARNN optimized for macro-F1 over-predicts the number of labels, indicating that tuning either version of this model is problematic. Next, we report the number of test instances for which a model did not make any prediction at a particular level ("No Prediction" in Table 5.3). This count is much lower for the TMM and THMM models, showing that hierarchical models can often make predictions at intermediate levels, even if the fine-grained class is unclear.

	Avg. Labels Predicted			**No Prediction**			**False Positives**	
level	**1**	**2**	**3**	**1**	**2**	**3**	**# inst.**	**hops**
gold	1.56	1.89	2.32	0	0	0	0.0	0.0
TwistBytes (Benites, 2019)	1.65	1.51	1.56	147	891	1,575	3,788	4.17±1.77
HARNN-orig (Huang et al., 2019)	1.36	1.17	1.02	116	1,134	2,466	2,341	4.08±1.79
HARNN (Huang et al., 2019)	2.29	2.62	2.82	0	12	148	6,380	4.12±1.79
flat-CNN (Li et al., 2018a)	1.31	1.45	1.67	512	512	512	4,198	4.38±1.69
TMM-CNN	1.75	1.98	2.01	**1**	**42**	228	5,236	4.22±1.69
THMM-CNN	1.68	1.92	2.04	5	55	232	5,282	4.17±1.69
flat-CLS (Lee and Hsiang, 2019)	1.26	1.39	1.61	570	570	570	3,916	4.28±1.72
TMM-CLS	1.59	1.68	1.71	13	125	476	4,114	4.19±1.72
THMM-CLS	1.61	1.84	2.03	8	66	**204**	5,046	4.22±1.68

Table 5.3: Analysis of coverage for USPTO dataset. *No Prediction*: number of test instances with no predicted labels at a given level. *False Positives (error analysis)*: average # hops between false positives and nearest true labels at the third level.

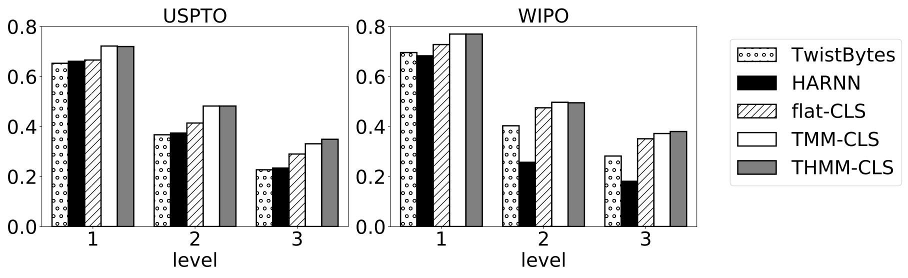


Figure 5.3: Classification performance: macro-average F1 by hierarchy level.

5.5.3 Error Analysis

In hierarchical classification tasks, misclassifications are not all the same. For example, in the case of subclass labels, misclassification of a non-sibling label is a worse problem than misclassification of any of the sibling labels. Thus, we analyze misclassifications in terms of the minimum number of hops between the predicted label and any of the true labels. A classifier with a shorter average minimum hop distance is considered better than one with a longer average minimum hop distance. With reference to the example CPC taxonomy shown in Figure 5.1, when considering A43B as a true label, predicting B41F is worse than predicting A43C because A43C is a sibling of A43B; i.e., both A43B and A43C are successors of A43, and therefore, have higher semantic proximity to A43C than B41F. In the CPC taxonomy tree, the labels A43C and B41F are two and six hops away from A43B, respectively.

Table 5.3 shows the average number of hops between false positives and true labels at level 3. The column titled **# inst.** denotes the number of test instances with at least one false positive label. In general, the incorrect predictions produced by all models seem to be at similar distances from the nearest true label and are usually within the correct section of the taxonomy. Again, the flat approach activates completely incorrect labels more often.

5.6 Conclusions

We propose a novel Transformer-based Multi-task Model (TMM) for hierarchical patent classification. The strength of our architecture lies in integrating the highly effective local-classifier-per-node concept of traditional hierarchical classification algorithms with a large-scale, pre-trained neural-transformer-based language model. This integration is computationally feasible due to our innovative multi-task-based architecture. We have shown that this model architecture strongly outperforms models developed in previous work on hierarchical text classification. Furthermore, it exhibits better coverage when predicting labels for test instances and improved performance when predicting less frequent labels. Furthermore, the hierarchical links in the THMM architecture improve performance for the infrequent labels at the lower level of the hierarchy.

For the models discussed in this chapter, the input text is limited to only the title and abstract fields of each patent. In the next chapter, we propose approaches for incorporating additional texts into the document representation, which are evaluated on the CPC classification task with THMM. Further, Chapter 7 enriches the document representation with additional CPC/IPC label information and evaluates it on multi-label PLS classification task with TMM.

Chapter 6

Efficient Neural Full-Text Patent Representations

In the previous chapter, we used SciBERT for text representation, which provides semantically rich representation. However, due to the quadratic time complexity associated with the full-attention mechanism (Vaswani et al., 2017), SciBERT takes limited input text, much shorter than a typical patent's length. In addition, transformer-based models with a longer context length than BERT, e.g., Longformer (Beltagy et al., 2020), CogLTX (Ding et al., 2020), and ToBERT (Pappagari et al., 2019), are found to be inefficient when evaluated on long text classification task by Park et al. (2022). Furthermore, parts of texts in a patent are redundant as the information on an invention is described at different levels of granularity across sections, which is further revealed with our analysis. We find that the abstracts and sentences are often duplicated within and across patents (see Section 4.6). Based on these aspects, in this chapter, we experiment with different document representation techniques, which efficiently generate a patent representation by selecting a subset of the most informative text elements (e.g., tokens, sentences, sections) from the full text of a patent.

Targeting patent classification as a use case, in Section 6.1, we underline the need for efficient patent classification techniques. In Section 6.2, we discuss the limitations of current state-of-the-art text representation techniques, particularly ones that are used for patent representation. Based on the identified research gaps, we propose document representation methods that first identify a subset of the most informative text elements and then generate a patent representation with the associated texts (Section 6.3). In Section 6.5, the proposed methods are evaluated and compared against the baselines on the CPC/IPC classification task using the experimental setup defined in Section 6.4.

6.1 Motivation and Contributions

As discussed in Section 5.5, the Transformer-based Multi-task Model (TMM) and its hierarchical variant the Transformer-based Hierarchical Multi-task Model (THMM) outperform neural (Li et al., 2018a; Lee and Hsiang, 2019; Huang et al., 2019) and non-neural (Benites, 2019) baselines in the CPC/IPC classification task (see Definition 2.1, page 18) by using SciBERT with concatenated text from title and abstract as input. However, although it is effective for shorter texts, the maximum sequence length constraint makes SciBERT non-ideal for patent classification tasks. The quadratic computational complexity restricts the input sequence length to 512 tokens, which is approximately 5% of the average length of a typical patent, which contains 12.5k tokens on average (see Figure 4.5, page 61).

Addressing the limitations of pre-trained language models such as BERT, some recent techniques, including Reformer (Kitaev et al., 2020), Longformer (Beltagy et al., 2020), BigBird (Zaheer et al., 2020), and CogLTX (Ding et al., 2020), introduce mechanisms for handling much longer input sequences. In the context of a classification task, most of these techniques divide a longer sequence into smaller text chunks, generate an embedding for each chunk, and aggregate the chunk embeddings to form a document-level representation. Even though the input for these models can be much longer than the input for SciBERT, these models fall short of accommodating the full texts of patents. For example, the maximum allowed input length for Longformer is 4096 tokens, which is approximately 33% of the average number of tokens in a typical patent. Furthermore, higher sequence lengths result in higher memory and computation costs. In a recent evaluation, Park et al. (2022) found that classifiers based on Longformer (Beltagy et al., 2020), CogLTX (Ding et al., 2020) or ToBERT (Pappagari et al., 2019) are highly inefficient, and in some cases their classification performance is worse than that of the BERT-based baseline.

The discussion so far indicates a trade-off between a classifier's efficiency and the maximum input length it permits. A classifier that accepts shorter input lengths is more efficient. However, it may miss any crucial semantic information that might be present in the ignored text. In contrast, incorporating additional text may lead to a performance gain but may, at the same time, lead to an increase in computation cost. Because of this, the majority of neural and non-neural patent classification approaches only consider title, abstract, and claims as patent fields, leaving out the more detailed and elaborate description field (Li et al., 2018a; Lee and Hsiang, 2019; Zaheer et al., 2020). Furthermore, the patent text is often truncated for efficiency reasons.

In addition to computational complexity, text redundancy is another factor that motivates our research on efficient patent representation methods that use limited yet informative text. The analysis in Section 4.6 shows that texts are often duplicated within and across patents. Furthermore, various patent sections describe the same invention at different levels of granularity and thus introduce redundancy across sections. Because text within a patent is often duplicated and redundant, we hypothesize that the total information within a patent is limited such that the core idea of an invention can be represented with a patent representation method that uses the most informative text chunks, a subset much smaller than the full texts of a patent.

Referring back to the conceptual model of the classification pipeline discussed in Section 3.2, we find that the patent document model (PDM) represents a typical patent. An instance of PDM can be mapped to an instance of the domain-independent document model (DIDM). The metadata, figures, citations, and content elements (representing patent text) are referred to as semantics elements, and all the semantic elements within a document are represented as U (see Equation 3.1, page 46). Since the information within a patent is limited, we can achieve optimal classification performance with limited texts taken from the most informative semantic elements in U by using an effective selection method.

The document representation techniques proposed in this chapter do not consider metadata for the CPC classification task, as they can introduce bias, be ambiguous, or be only partially known. For example, the inventors' and assignees' names are often found to have discrepancies in filings and are often ambiguous (Haak et al., 2012). Additionally, crucial citations are typically unknown during the filing of the application and later added by the patent examiner (Alcácer et al., 2009). Also, we do not consider figures in the current scope of the thesis.

Ignoring metadata, citations, and figures, we are left with four types of content elements, field, passage, sentence, and token, which are arranged at four levels of granularity within the hierarchical structure of a DIDM (see Figure 3.5, page 45). In this chapter, we evaluate document representation techniques that rank semantic elements according to their importance, select a subset of the most informative semantic elements, and generate a document representation using the corresponding text. Later in this chapter, our experiments evaluate the proposed document representation techniques and validate the hypothesis (Section 6.5). We pose the following questions to aggregate a document representation from limited yet informative semantic elements.

- Given a patent document as a collection of semantic elements arranged at different levels of granularity (e.g., tokens, sentences, paragraphs, and sections), at which level of granularity should we select them to achieve better performance?

- Given the level of granularity for selecting semantic elements, how should we rank them effectively to achieve the desired performance with a minimal subset of elements?

- Given the level of granularity and a technique to rank semantic elements, how many semantic elements should we select to obtain the desired classification performance, such that adding more semantic elements only achieves a marginal gain or even a decrease in classification performance?

This chapter evaluates document representation methods by aggregating information at three levels of granularity: field, sentence, and token. Here, fields correspond to the sections and subsections within a patent. Passages are not considered for document representation because passages in patents can be extremely long and often incorrectly delimited.

Further, depending on the data source, passage boundaries may not be explicitly marked, thus making it challenging to identify them. All the experiments in this chapter were performed on the CPC classification task with the USPTO-70k dataset (see Section 4.3).

We evaluate document representation methods using the texts corresponding to the patent fields by considering patent fields individually or in combination. The field embeddings are combined in order of the informativeness of the fields, which is determined based on the classification accuracy achieved using the field text.

When considering sentences as a source of semantic information, the subset of the most informative sentences is determined based on sentence scores assigned by a sentence ranker. We propose a novel sentence ranker pos-field_label-dense that computes sentence score based on the dot-product similarity between the sentence embedding and label description embeddings, the importance of the field to which a sentence belongs, and the position of the sentence within the field. The classification performance of the top-ranked sentences is compared against the sentence rankers associated with extractive summarization techniques, PacSum (Zheng and Lapata, 2019) and RankSum (Joshi et al., 2022). Section 6.3 provides details on different sentence ranking methods.

Patent representations are generated using neural and non-neural techniques for the texts of fields and sentences. In the case of neural methods, we apply the SciBERT model, fine-tuned on the CPC classification task in Section 5.5, for generating embeddings for texts corresponding to patent fields and sentences without further fine-tuning. Since the SciBERT parameters are not fine-tuned in the process, the training time is much less as compared to the classifiers from Chapter 5, thus making the training process efficient. A non-neural representation is generated using TF-IDF (see Section 2.4.1). The neural and non-neural representations are then provided as input to the THMM (see Section 5.3) and TwistBytes (Benites, 2019) models, respectively.

In addition, we evaluate representation by selecting the most informative key phrases from a document. First, we extract a set of key phrases using TextRank (Mihalcea and Tarau, 2004), then concatenate the top key phrases and use the concatenated text to generate a document representation Mihalcea and Hassan (2005).

By evaluating the performance of document representation techniques that select the most informative semantic elements across granularity levels, we find that optimal performance can generally be achieved with a subset of the most informative semantic elements. Further, in line with the findings of Xue and Croft (2009), we find that brief-summary is the most informative field. However, it should be noted that Xue and Croft (2009) targeted the patent retrieval task using non-neural methods. Next, by comparing the neural and non-neural document representations considering patent fields, sentences, and tokens, we find that neural representation combining the SciBERT-based field embeddings using vector summation achieves the best performance. Further, the pos-field_label-dense sentence ranker proposed in this chapter performs better or comparable to the top baseline sentence ranking methods. The main contributions of this chapter are thus as follows:

- We propose a **novel approach that combines patent field embeddings using vector summation** and performs better than the neural and non-neural baseline, including the best-performing classifier from Chapter 5 that apply THMM with the SciBERT representation for concatenated text from title and abstract as input.
- We propose a **novel sentence ranker for efficient patent classification** that ranks a sentence based on its semantic similarity to label descriptions, the position of the sentence within the field, and the importance of the field to which the sentence belongs. Our analysis shows that the extractive summarization techniques are effective in selecting informative texts for neural representation in the context of patent classification.
- We demonstrate that **providing informative texts** to a classifier can **improve the classification accuracy** on the CPC classification task without further fine-tuning the pre-trained language model.

6.2 Related Work

While we mainly focused on patent classification models in Section 5.2, in this section, we discuss methods used to represent text in key prior works, particularly in the context of patent classification. Depending on text representation methods, we categorize the prior patent classification works into three groups: bag-of-words (BoW), pre-transformer neural methods, and transformer-based neural methods.

BoW methods compute a feature vector using term statistics, for example, term count (Larkey, 1999) and ignore the sequence information in a text. Most BoW-based patent classification techniques only use the abstract to compute the feature vector (Krier and Zaccà, 2002; D'hondt et al., 2013; Tran and Kavuluru, 2017), with the exception of Benites et al. (2018), which use the full text of a patent.

Comparing the application of different text representation methods to various text classification tasks, Galke and Scherp (2022) identify TF-IDF-based classifiers as a strong baseline (see Section 2.4.1). However, most recent deep learning methods for patent classification (Li et al., 2018a; Lee and Hsiang, 2019; Zaheer et al., 2020) are not compared to a TF-IDF-based baseline, despite the fact that it is particularly important for patents as it can accommodate the full texts of patents. The evaluation performed in this chapter includes a comparison with a TF-IDF-based baseline that takes the full texts of patents as input.

Neural patent classification methods developed in the pre-transformer era typically represent the patent text as a sequence of non-contextualized embeddings, e.g., word2vec (Mikolov et al., 2013a), and generate a sequence representation using CNNs (Li et al., 2018a; Abdelgawad et al., 2019), LSTM (Shalaby et al., 2018), GRUs (Risch and Krestel, 2019), or a combination of CNNs with BiLSTM (Hu et al., 2018). The text from the patents' title, abstract, and claims is the primary choice for representation, with some exceptions (Abdelgawad et al., 2019; Hu et al., 2018). Abdelgawad et al. (2019) used the first

1,500 tokens from patents for representation, whereas Hu et al. (2018) considered the first 150 tokens each from the abstract, claims, and description sections.

The third category of methods includes patent classification techniques developed following the phenomenal success of BERT (Devlin et al., 2019) and other transformer-based language models (Liu et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020), which use self-attention (Vaswani et al., 2017) or sparse attention (Child et al., 2019) to represent tokens in the sequence. In the context of the CPC/IPC classification task, the work of Lee and Hsiang (2019) was the first to use BERT for the patent classification task, achieving better performance than Li et al. (2018a). Althammer et al. (2021a) performed domain-adaptive pre-training of BERT and SciBERT models over patent text and found that SciBERT performed better for the CPC classification task. Although effective for patent classification, SciBERT can accommodate only 512 tokens as input. To address this challenge, Zaheer et al. (2020) propose BigBird, a sparse attention-based language model, which can take 4096 tokens as input. Their evaluation of the CPC classification task showed that BigBird performs better than RoBERTa (Liu et al., 2019). Comparing multiple transformer-based language models, Roudsari et al. (2022) found that the XLNet (Yang et al., 2020) performs best on patent classification tasks.

In addition to the underlying transformer model, another important consideration in patent classification is which patent fields should be used as input to the model. Li et al. (2018a) evaluated the use of title and abstract or only claims as input to BERT, finding that using only claims as input resulted in better performance. Consequently, Althammer et al. (2021a) also used claims in their experiments. Zaheer et al. (2020) provided the concatenated text of the title, abstract, and claims as input to the language model. Due to non-standard terminology, claims are often interpreted with reference to the patent's description. Therefore, unlike previous works, we experiment with the description section in addition to the title, abstract, and claims sections. We differ from the past transformer-based techniques on patent classification in the following aspects. Firstly, we make use of transfer learning by using the SciBERT model fine-tuned on the CPC classification task in Section 5.5. Secondly, we consider the full text of a patent as a candidate when selecting sentences by using the sentence rankers and generating a patent representation. Li et al. (2022) apply contrastive learning to learn patent embedding from claims text.

Apart from the text representation methods used for patent classification, in the following, we review some works that aim to represent text much longer than the maximum length limit for BERT. Yang et al. (2016) propose a document representation technique that uses a hierarchical attention mechanism, first encoding the sentence-level information and then aggregating it at the document level, with both levels using BiGRUs as an encoder. The representations for sentences and documents are generated by aggregating words and sentence embeddings. This is done using context vectors, against which the attention weights are computed with the assumption that important words and sentences will be more similar to the context vector. Longformer (Beltagy et al., 2020) was one of the first models to use sparse attention to represent longer sequences. It uses the attention mechanisms at two levels. First, within a fixed-size window, all tokens attend to each other. Second, a few selected tokens, one in each window, attend to each other globally, thus capturing long-range dependencies. BigBird (Zaheer et al., 2020) differs slightly from Longformer as it enables a few more randomly selected tokens that mutually attend to each other as well as the local and global attention heads. With a special focus on long-text classification, Dai et al. (2022) propose an approach that first generates segment embeddings using RoBERTa, then passes these through two stacked transformer layers, and finally aggregates them to generate a representation with a max pooling operation. Chalkidis et al. (2022) propose a model architecture with several layers of segment-wise and cross-segment transformer blocks, in which a segment-wise block processes each segment independently, and a cross-segment block computes attention between the segment-level tokens, i.e., the [CLS] token of each segment, across segments in the previous layer.

Dong et al. (2023) provide a detailed survey of long-text representation methods in the context of four NLP tasks, including text classification. They find that most methods perform three key steps— truncation, chunking, and selection—to determine the input for text representation methods. The methods discussed above use truncation or chunking to decide the input sequence. However, in both cases, the maximum text limit allowed is much less than the average length of a typical patent text. Further, our evaluation in Section 4.6 demonstrates that the patent text is often duplicated. For these reasons, we explore methods of identifying and selecting the most informative text elements.

Over the years, several techniques have been evaluated for the selection of key text elements from a document as a pre-step for classification. A study by Ker and Chen (2000) was one of the first to apply text summarization strategies in the context of text classification. Kolcz et al. (2001) experiment with several text summarization strategies, for example, by considering the title, the first paragraph, and the paragraph with the most keywords. Ko et al. (2002) select sentences based on their similarity with the document's title and the importance of terms within the sentence text. Following their work on TextRank (Mihalcea and Tarau, 2004), Mihalcea and Hassan (2005) represent a document as a graph with sentences as nodes. They use page-rank-based centrality criteria to identify the key sentences in a document and use them to generate a TF-IDF feature vector. Over the years, several other works have applied text summarization as a feature selection strategy when used with a term-count-based feature vector (Hulth and Megyesi, 2006; Jiang et al., 2009; Anguiano-Hernández et al., 2010). In a related work, de Souza et al. (2021) use abstractive and extractive summarization to generate associated subgroup labels' descriptions from input patent text. They evaluate the output using the ROUGE set of metrics (Lin, 2004) to check whether the model can generate a text matching the CPC/IPC label descriptions.

Employing the concept of working memory (Baddeley, 1983), Ding et al. (2020) propose CogLTX, a BERT-based method, which selects key text chunks from the document for representation. In their analysis, Park et al. (2022) found CogLTX and Longformer (Beltagy et al., 2020) to be highly inefficient in terms of long-text classification benchmarks, compared to BERT and simple BERT-based baselines that combine the embedding of the first 512 tokens to that of keywords selected using TextRank (Mihalcea and Tarau, 2004).

Based on duplication analysis of patent texts (see Section 4.6), the evaluation performed by Park et al. (2022), and the success of extractive summarization as a feature extraction method, we propose methods for the selection of the most informative text elements and use the corresponding texts for neural and non-neural patent representation. We propose a novel sentence ranker that ranks sentences based on their position within patent fields, the informativeness of the corresponding patent fields as determined based on the classification performance of fields' texts, and the semantic similarity between sentences and label descriptions. We demonstrate the applicability of extractive summarization techniques for neural representation in the context of patent classification. The best performance in the CPC classification task is achieved with our proposed approach of combining SciBERT-based field embeddings using vector summation. In this chapter, we take a data-driven approach by employing a simple model architecture, using the most informative texts for patent representation, and employing the SciBERT model that is fine-tuned on the CPC classification task in the previous chapter without performing any further fine-tuning. Next, we describe the various patent representation methods proposed in this chapter.

6.3 Patent Representations

In Section 6.1, we hypothesized that optimal classification performance could be reached by using the text corresponding to the most informative semantic elements, e.g., fields, sentences, and tokens. Based on this hypothesis, here we define techniques for ranking semantic elements and propose document representation methods that are evaluated on the CPC classification task, as detailed in Section 6.5.

The classification pipeline discussed in Section 3.2 maps an instance of patent document model (PDM) to an instance of domain-independent document model (DIDM), which is further mapped to a data model instance that is specific to the document representation method (DocRepDM). The document representation methods described in this section take an instance of $\mathrm{DocRepDM\_Chunks}$ as input (see Section 3.2.4). A mapper function maps a DIDM instance to the instance of $\mathrm{DocRepDM\_Chunks}$ by selecting the texts from fields or sentences. $\mathrm{DocRepDM\_Chunks}$ has chunks as an attribute, which is a list of Chunk instances, each of which has text, $\mathrm{max\_length}$, and $\mathrm{position}$ as attributes. In the current context, text corresponds to the text for a field or a sentence, $\mathrm{max\_length}$ is the length at which the text is truncated to generate an input sequence, and $\mathrm{position}$ is the relative position of the text chunk among its siblings, e.g., position of a sentence among other sentences within a field or a document. Next, we look into the details of the proposed document representation methods.

6.3.1 Field-Based Representation

As discussed, given a set of fields F within a document, the task is to determine the field subset $F^* \subseteq F$ that achieves optimal classification performance in the CPC classification task. For neural methods, the field embeddings are pre-computed using the SciBERT model, which has already been fine-tuned on the CPC classification task (see Definition 2.1, page 18) in Chapter 5. Previous works showed that in the context of the CPC/IPC classification task, the initial text of a field is more informative than the text in the later part (Guyot et al., 2010; Wu et al., 2010; Verberne and D'hondt, 2011; Li et al., 2018a). Therefore, we truncate the field text to the maximum sequence length value before generating a field embedding. The field embeddings are generated using the process defined in Section 2.4.3. The field text is truncated to a maximum sequence length of 512 tokens and passed through the SciBERT model. The final hidden state corresponding to the [CLS] token is taken as field embedding. Next, the field embeddings are evaluated separately or in combination for the CPC classification task. The relative importance of a field is determined based on the performance of the corresponding field embeddings in the CPC classification task. Finally, the field embeddings are combined in decreasing order of importance using vector summation or concatenation, and the resultant vector is evaluated on the CPC classification task. Such a representation is more efficient than the text representation method used in the previous chapter since the SciBERT weights are not fine-tuned. A non-neural representation is generated using TF-IDF with concatenated fields' texts as input. The SciBERT-based embedding for a patent and its TF-IDF vector are evaluated in combination with THMM and TwistBytes (Benites, 2019), respectively.

6.3.2 Sentence-Based Representation

Given a set of sentences S, we aim to determine a subset S^* which consists of the most informative sentences, such that $S^* \subseteq S$ and $p \leq n$. Since patent documents contain redundant information, we assume that optimal classification performance can be achieved with a p value much lower than the average number of sentences within patents.

One approach to the determination of S^* is to define a sentence ranker that scores and ranks the sentences in S. The sentence ranker assigns a score to each sentence based on its importance, and the top p sentences with the highest scores are selected as input for the classifier. This approach allows us to focus on the most informative sentences and disregard redundant and uninformative texts.

The sentence selection step discussed above is similar to the sentence extraction step in an extractive summarization task that assigns a score to each sentence and selects the top-ranked sentences as a document summary (El-Kassas et al., 2021). Leveraging this commonality with extractive summarization techniques and drawing inspiration from the prior work on using extractive summarization for feature extraction (Mihalcea and Hassan, 2005; Hulth and Megyesi, 2006; Anguiano-Hernández et al., 2010), we experiment with sentence rankers based on the unsupervised extractive summarization algorithms PacSum (Zheng and Lapata, 2019) and RankSum (Joshi et al., 2022). These algorithms are well-suited for the patent classification task as they can handle extremely long documents, are efficient, and do not alter the sentence text during extraction.

Furthermore, unlike Sharma et al. (2019), we opt for unsupervised methods instead of using $\mathrm{abstracts}$ as patent summaries for two reasons. First, as discussed in Chapter 4, abstracts are often written less cautiously than other sections and are prone to duplication, which may result in the loss of fine-grained invention details (see Section 4.6). Second, we have already experimented with abstracts and now seek more information from other patent fields—including the description field, which is often ignored in patent classification tasks. Once we determine the set S^*, we can generate a document representation using either neural or non-neural methods.

When generating a document representation using *neural* methods, we first map each sentence s_i in S^* to an embedding $\boldsymbol{s_i}$ using a process similar to the generation of a field embedding (see Section 2.4.3). We then use the vector mean of the sentence embeddings as the document representation $\boldsymbol{x}^{S^*}$. The resulting document representation is provided as input to the THMM model. In the current setting, the SciBERT parameters are also fixed, and the sentence embeddings are pre-computed, resulting in an efficient training process.

In a *non-neural* setting, we concatenate sentences from S^* to generate a text for each document in the training set. Then, we compute the term statistics for the TF-IDF model and use it to generate a feature vector for each document in the dataset. Finally, we provide these feature vectors as input to TwistBytes (Benites, 2019).

We evaluate a diverse set of sentence rankers that capture various notions of sentence importance. The pac-sum ranker is based on PacSum (Zheng and Lapata, 2019), while the other four evaluated baseline rankers (pos-doc, saliency, keyword, and lda) are based on RankSum (Joshi et al., 2022). We propose two novel sentence rankers: pos-field and label-sim. The pos-field ranker ranks a sentence based on its position within a field and the importance of the field. In contrast, the label-sim ranker ranks a sentence based on the semantic similarity between the sentence and label descriptions. We find that a marginal gain in performance is achieved with a hybrid ranker pos-field_label-dense that combines the sentence scores from the pos-field and label-sim rankers. Details on the various sentence rankers are provided below.

Semantic Sentence Graph (pac-sum). Zheng and Lapata (2019) propose a sentence ranker that represents each document as a semantic graph in which the nodes correspond to the sentences in the document. To achieve this, each sentence is first represented as an embedding using a pre-trained language model that is fine-tuned for a sentence similarity task. In the context of patent classification, we generate sentence embeddings using a SciBERT model, which has been fine-tuned for the CPC classification task (as discussed in Section 5.3). Specifically, given a pair of sentences s_i and s_j, their corresponding embeddings are denoted as $\boldsymbol{s_i}$ and $\boldsymbol{s_j}$, respectively. The similarity score between the two sentences, denoted as e_{ij}, is computed by taking the dot product of their embeddings as follows:

$$e_{ij} := \boldsymbol{s_i}^T \boldsymbol{s_j} \tag{6.1}$$

The similarity scores for all the sentence pairs are represented as a similarity matrix $\boldsymbol{E}$. Further, the similarity scores are normalized as shown in Equation 6.2, and finally, the negative values in matrix $\boldsymbol{E}$ are set to 0. The hyper-parameter β controls the threshold for the similarity score value.

$$e_{ij} := e_{ij} - [min(\boldsymbol{E}) - \beta(max(\boldsymbol{E}) - min(\boldsymbol{E}))] \tag{6.2}$$

The position-augmented degree centrality for a sentence s_i is computed as follows (Zheng and Lapata, 2019):

$$centrality(s_i) := \lambda_1 \sum_{1 \le j < i} e_{ij} + \lambda_2 \sum_{i < j \le n} e_{ij} \tag{6.3}$$

For a given sentence s_i, the contributions to the degree centrality from the sentences appearing before and after it are weighted with different weighing parameters λ_1 and λ_2, respectively. Next, we describe the four sentence rankers associated with RankSum (Joshi et al., 2022).

Position of a Sentence within a Document (pos-doc)**.** Using the position-based ranking scheme from RankSum (Joshi et al., 2022), the sentences appearing earlier in a patent are ranked higher. With pos-doc, the sentence set S^* is determined by selecting the first p sentences from the document.

Sentence Saliency (saliency)**.** The saliency-based sentence ranker proposed by Joshi et al. (2022) considers a document as a cluster of sentence embeddings and assumes that the cluster centroid represents the core idea of the document. Accordingly, the importance of a sentence is determined based on the shift of the centroid upon removing the sentence. Specifically, the larger the shift of the centroid, the more important the sentence is deemed to be.

Given a set S of sentences for a document, the centroid vector $\boldsymbol{c}_S$ is computed by taking the mean of all the sentence embeddings corresponding to the sentences in S. For a sentence s_j, the centroid vector $\boldsymbol{c}_{S\setminus\{s_j\}}$ is computed by first removing the sentence s_j from S and then determining the mean of all the embeddings corresponding to sentences in $S \setminus \{s_j\}$. The sentence score is computed by first taking the dot product between $\boldsymbol{c}_S$ and $\boldsymbol{c}_{S\setminus\{s_j\}}$, and then scaling it to a value between 0 and 1. Finally, the resulting similarity value is subtracted from 1 to obtain the sentence score. As previously mentioned, the saliency ranker assumes that a sentence is more important if the similarity between $\boldsymbol{c}_{S\setminus\{s_j\}}$ and $\boldsymbol{c}_S$ is lower.

Keyword Count for a Sentence (keyword)**.** Joshi et al. (2022) propose a keyword-based sentence ranker that builds on the assumption that key phrase extraction methods such as TextRank (Mihalcea and Tarau, 2004) identify the most important key phrases and rank them in terms of their importance. Specifically, the ranker computes a score for each sentence based on the number of key phrases it contains. Sentences with higher key phrase counts are ranked higher. In cases where two sentences have the same number of key phrases, the sentence rank is determined based on their relative positions within the document, with sentences appearing earlier in the document being ranked higher.

LDA-based Representation for Sentences and Documents (lda). Joshi et al. (2022) propose a sentence ranker based on Latent Dirichlet Allocation (Blei et al., 2003, LDA) that first trains an LDA model on the training set, then represents each document and sentence as LDA topic vectors, and finally computes the Euclidean distance between the sentence and document vectors to determine sentence score. To compute a document vector, the tokenized text of the document is passed through the LDA model. For each sentence, a sentence vector is computed by taking the mean of the topic vectors corresponding to the words within the sentence. The sentence score is determined by calculating the Euclidean distance between the sentence and document vectors, then scaling the distance to a value between 0 and 1, and subtracting the scaled distance value from 1. The LDA-based ranker assumes that the smaller the distance between a sentence and the document vector, the more important the sentence is. Next, we describe the novel sentence rankers proposed in this chapter in the context of patent classification.

Position within a Field and Field Importance (pos-field). As discussed, the pos-doc ranker is based on the assumption that sentences appearing earlier in a document should be ranked higher in importance. Such a ranking scheme assigns a lower rank to sentences appearing later in the document, even if they are located within an important field. For example, even though we found that the claims field is more informative than the detail-desc field (Section 6.5.1), sentences within claims are ranked lower by the pos-doc ranker as they appear later in a patent. As shown in Figure 6.4f, the claims field is almost ignored when selecting the top-50 sentences using the pos-doc ranker. To boost the scores for sentences belonging to more important fields, we introduce a novel sentence ranker based on two assumptions:

- Given a sentence pair (s_i, s_j), a sentence appearing earlier in a field is ranked higher than the one appearing later, i.e., $r^{pos\_field}(s_i) > r^{pos\_field}(s_j)$, if $s_i.pos\_parent < s_j.pos\_parent$. The method $r^{pos\_field}(.)$ gives the rank of a sentence, and $\mathrm{pos\_parent}$ represents the position of a sentence within the field as a parent (see Figure 3.5, page 45). Since we do not consider passages, sentences have fields as parents. The condition is valid for sentences across fields. For example, a sentence at position one in the abstract is ranked higher than those at position two or higher in the abstract or any other fields.

- In cases where $s_i.pos\_parent = s_j.pos\_parent$, sentence rank is determined based on the field importance score, such that sentences appearing in a more important field are ranked higher. The method $fi(.)$ takes a sentence as input and provides a field importance score for the corresponding field to which a sentence belongs, such that the field that performs better in the classification task is assigned a higher importance score. Thus, for a condition $s_i.pos\_parent = s_j.pos\_parent \wedge fi(s_i) > fi(s_j)$, the rank for sentence s_i is higher than the rank for sentence s_j. For example, given that the $\mathrm{brief\text{-}summary}$ has higher importance than the $\mathrm{abstract}$, the rank for a sentence at position two in the $\mathrm{brief\text{-}summary}$ is higher than for the sentence at position two in the $\mathrm{abstract}$.

Sentence and Label Similarity (label-sim)**.** The rankers discussed so far use sentence position and relevance to the document's topic as ranking criteria. Targeting the text summarization task, Peyrard (2019) identifies informativeness as another crucial criterion for sentence importance. A label description describing an important domain concept is associated with each CPC label. Thus, the informativeness of a sentence can be determined based on its similarity to label descriptions.

The label-similarity ranker assumes that a sentence is informative if it is semantically similar to any of the label descriptions. Earlier in the section, we discussed steps to compute embeddings for sentences in S^*, which are represented as a matrix $\boldsymbol{S}^*$ of size $|S^*| \times dim_{SciBERT}$, where $dim_{SciBERT}$ is the size of a sequence embedding computed with SciBERT. The label embeddings are computed by passing the label description through SciBERT as an input sequence and taking the last hidden state of the [CLS] token as a sequence embedding. In this case, we use a SciBERT model fine-tuned for the CPC classification task, as detailed in Section 5.3. In addition, since the subgroup labels, i.e., levels five and below, represent concepts at the lowest level of the hierarchy, we compute a label embedding for all subgroup labels, represented as a matrix $\boldsymbol{C}_{sg}$ with size $|C_{sg}| \times dim_{SciBERT}$ where C_{sg} is the set containing all CPC subgroup labels. The label-similarity scores are computed as follows:

$$\boldsymbol{M} := \boldsymbol{S}^* \boldsymbol{C}_{sg}^T \tag{6.4}$$

The value m_{ij} at the i^{th} row and the j^{th} column in $\boldsymbol{M}$ corresponds to the similarity score between the i^{th} sentence and the j^{th} label. For the i^{th} sentence, the label-sim score is equal to the maximum value in the i^{th} row, i.e., the maximum similarity value of the sentence to any of the label descriptions. Thus, when using the label-sim ranker, a sentence is ranked higher if it has higher semantic similarity to any of the label descriptions.

Random Ranker (random-ranker). To understand how well a sentence ranker can identify important sentences, we compare each ranker to a random ranker (random-ranker) as a baseline. A random-ranker selects sentences at random from a document. Since most of the sentences within a patent contain information about the proposed invention, we assume that a random selection will also provide relevant sentences sufficient for classification. Thus, an effective sentence ranking algorithm should perform better than a random-ranker.

Hybrid Ranker. Our analysis indicates that label-dense and pos-field select sentences from different parts of patents and perform better than the random-ranker (Section 6.5). We combine these two rankers by assigning equal weights to scores from pos-field and label-dense, which we refer to as pos-field_label-dense. In addition, we combine pos-doc and label-dense, which we refer to as pos-doc_label-dense.

6.3.3 Token-Based Representation

Like field- and sentence-based representation learning, a token-based representation is generated by identifying the most important tokens within a document and then generating a document representation using a TF-IDF model. We assume that tokens appearing in key phrases are important and indicate the document's topic. Similar to the keyword ranker, the key phrases within a document are identified using TextRank (Mihalcea and Tarau, 2004). For each document in the training, validation, and test sets, the key phrases are extracted using TextRank. Comparing BoW representation methods, Galke and Scherp (2022) find that the TF-IDF weighted representation works better than a representation generated by averaging Glove-based token embeddings (Pennington et al., 2014). Next, a text is generated for each document by sorting key phrases in order of importance and then concatenating the associated text. An analysis is performed by tokenizing the concatenated text and selecting k tokens provided as input to a TF-IDF model. The term statistics for the TF-IDF model are determined using text corresponding to the documents in the training set. The TF-IDF model is then used to generate a vector for each document within the training, validation, and test sets. The TF-IDF vector is provided as input to TwistBytes (Benites, 2019). The evaluation results are discussed in Section 6.5. In this case, we experiment only with the non-neural TF-IDF representation and the TwistBytes model since the sequence information is unavailable.

6.4 Experimental Setup

Dataset and Evaluation Metrics. The evaluations are performed using the USPTO-70k dataset that contains about 50,000 training instances and 10,000 instances each for validation and test set, respectively (see Section 4.3). Each instance has six text fields: title, abstract, claims, brief-summary, detail-desc, and fig-desc. Further, each instance is labeled with CPC labels as target classes. The document representation techniques are evaluated on the CPC classification task using the hierarchical evaluation metrics defined in Section 2.7. In the case of hierarchical evaluation metrics, the true label set consists of labels across the hierarchy. To evaluate a partially correct prediction, the ancestors of the predicted labels are also added to the predicted label set before calculating the evaluation score. Specifically, we compute the macro and micro average scores for hierarchical precision, recall, and F1. The micro-average score is computed over all the instances in the test set and provides an overall estimate of a model's performance. The macro-average score is an average of the performance over all the labels, thus evaluating a model's ability to perform better in a class-imbalance scenario.

Hyperparameters. Below, we define the hyperparameters for the text representation methods, sentence rankers, and classification models.

Text Representation. We use SciBERT to pre-compute the field and sentence embeddings of dimensionality 768. The SciBERT model has already been fine-tuned for the CPC classification task with title and $\mathrm{abstract}$ as input, as discussed in Section 5.5. The texts for fields and sentences are represented as TF-IDF vectors of dimensionality 70k. Most patent sections exceed the 512-token limit imposed by SciBERT (see Figure 4.5, page 4.5). The maximum input length is set to 512 tokens for all sections where this is the case. In patent sections that only require shorter sequences, the maximum input length is reduced for efficiency reasons. For the title and $\mathrm{abstract}$ sections, for example, we use the maximum sequence length values of 64 and 256, respectively. The details on the process of generating sequence embedding from a BERT-based model are provided in Section 2.4.3.

Sentence Rankers. Following Zheng and Lapata (2019), the hyperparameters for the $\mathrm{pac\text{-}sum}$ algorithm, λ_1, λ_2, and β, are set to values of -2, 1, and 0.6, respectively. For the lda ranker, we train an LDA-topic model with a topic size of 100. The label embeddings for the $\mathrm{label\text{-}sim}$ ranker are computed using the SciBERT model that is fine-tuned for the CPC classification task as described in Section 5.5, and each of the label embeddings is of dimension 768.

Classification Model. The experiments are performed with THMM using the set of hyperparameters defined in Section 5.4.3. Since SciBERT model weights are not fine-tuned during training, a learning rate of 10^{-3} is used. All models are trained for a maximum of 100 epochs with early stopping, i.e., the training stops if the macro-F1 for the validation set does not increase for 10 epochs.

Baselines. Below, we define the baselines against which the proposed classifiers are compared.

Non-Neural. Unlike the approach in Chapter 5, where the TwistBytes model (see Section 2.6.3) is used with just the title and $\mathrm{abstract}$ as input, here analysis is performed by adding field texts in order of their importance. We use the same parameter values defined by Benites (2019): a decision function threshold value of -0.25 and TF-IDF vectors of dimension 70k.

Neural. As a *neural* baseline, we chose the best-performing THMM model from Section 5.5, i.e., *THMM with* $e_f(t + a)$. Here, the method $e_f(.)$ represents the process of generating an embedding for an input sequence, and the subscript "*f*" denotes that the model weights are fine-tuned during training (see Section 2.4.3). In addition, we compare different document representations to one generated from all the field embeddings (*THMM with* $\boldsymbol{x}^F$) and sentence embeddings (*THMM with* $\boldsymbol{x}^S$).

Runtime. The TwistBytes model takes four to six hours to train when provided with the full texts of patents. Since the sentence and field embeddings are pre-computed, only the parameters corresponding to the classification model, i.e., THMM, are learned during training. Here, the training process takes 20 to 24 hours, as compared to 300 hours when fine-tuning the SciBERT parameters during training. Each of the models is trained on a single Nvidia Tesla V100 GPU. The early stopping is employed if the macro-F1 score on the validation set does not improve for seven epochs.

6.5 Results

Working with the hypothesis that a limited number of the most informative semantic elements are sufficient for optimal classification performance, we described document representation techniques in Section 6.3. Here, we evaluate the performance of the proposed techniques on the CPC classification task.

6.5.1 Performance for Field-Based Representations

In Section 6.3.1, we discussed methods for generating field embeddings and aggregating them to a document representation. As the first step, the relative importance of field embeddings is determined based on the performance of field text when evaluated on the CPC classification task. Next, a representation is generated by combining field embeddings using vector summation and concatenation.

Informativeness of Individual Fields. To assess the contribution of different fields to informativeness with regard to document classification, we use one field at a time, generate a document representation, and evaluate it on the CPC classification task. We find that the brief-summary is the most informative field in terms of overall performance, showing a high score across metrics and clearly outperforming models leveraging abstract and claims (see top-most part of Table 6.1). Unlike detail-desc, brief-summary often includes a summary of the invention and precise details on its technical field, which may be a possible reason for its higher informativeness compared to other patent fields. As title contains a few terms describing an invention with absolute brevity, a document representation based on title can identify some labels with high precision but only has a low recall. A similar conclusion can be drawn for fig-desc, as it contains particular domain-related terms. Further, the claims field has legal implications and is specific to an invention. In contrast, abstracts are not legally binding and cannot be challenged with litigation. Thus, abstracts are often drafted broadly and imprecisely, partially explaining the limited usefulness of this section for classification tasks like ours; abstract results in a high recall score but lower precision.

Performance when Combining Fields. We aim to achieve optimal classification performance with a subset of fields, such that adding more fields results in only a marginal gain or even a decrease in performance. The fields are combined according to their informativeness to generate a document representation (Table 6.1). With both vector summation and concatenation, we see an increase in the micro-F1 score when adding more field embeddings; however, the gain is marginal with concatenation. When comparing macro-F1 scores, a marginal gain (for summation) or decrease in performance (for concatenation) can be seen when adding additional embeddings to $\boldsymbol{bs}$ and $\boldsymbol{a}$. For summation, the relative gain in macro-F1 resulting from the incorporation of $\boldsymbol{a}$ to $\boldsymbol{bs}$ is 4.1%, whereas when further incorporating the additional four field embeddings the relative increase in macro-F1 is just 2.1% (see bottom-most part of Table 6.1). A similar conclusion can be drawn for TwistBytes, in which case classification performance improves when the text from $\mathrm{abstract}$ is added, but adding more fields decreases macro-F1. Furthermore, for TwistBytes, the micro-F1 score gradually increases when text from additional fields is added. These observations lead us to conclude that optimal macro-F1 can be reached with $\mathrm{brief\text{-}summary}$ and $\mathrm{abstract}$, such that adding more patent fields may only lead to a marginal gain or to a decrease in the macro-F1 score.

Comparison of Best-Performing Configuration to Baseline Systems. First, we note that TwistBytes using TF-IDF-based representations of the complete document text has a higher micro-F1 score than the THMM with $e_f(t + a)$), i.e., the best-performing model from Chapter 5. This demonstrates that there is relevant information in the additional text and motivates us to combine multi-field embedding into a single *neural* document representation using an effective aggregation technique. Table 6.1 shows the evaluation results comparing the field-level document representation to the baselines (*TwistBytes with* full-text and *THMM with* $e_f(t+a)$). The proposed approach of combining the field embeddings with vector summation (*THMM with* $\boldsymbol{t} \oplus \boldsymbol{a} \oplus \boldsymbol{cl} \oplus \boldsymbol{dd} \oplus \boldsymbol{bs} \oplus \boldsymbol{fd}$) outperforms both baselines in terms of macro-F1 and micro-F1 scores. When comparing to *THMM with* $e_f(t+a)$, we see much improvement in precision with a slight dip in recall. The proposed approach performs better across all three micro-average scores, i.e., precision, recall, and F1.

Performance by Label Frequency. Table 6.2 shows the results of our best performing approach (*THMM with* $\boldsymbol{t} \oplus \boldsymbol{a} \oplus \boldsymbol{cl} \oplus \boldsymbol{dd} \oplus \boldsymbol{bs} \oplus \boldsymbol{fd}$) and baselines for different frequency-based groups (Table 4.2). We find that our approach works better overall, and in particular, for the most infrequent label set, i.e., for a set consisting of labels with counts lower than 10, the macro-F1 is 44% better than for *THMM with* $e_f(t + a)$. The performance of the non-neural TwistBytes model is strong for more frequent labels but poor for infrequent labels.

Analysis of Field Embeddings by CPC Level, Label Frequency, and Domain. Here, we report a fine-grained analysis of model performance for three label groupings (see Table 4.2, page 60), focusing on macro-F1. The results are depicted in Figure 6.1. Overall, models using the $\mathrm{brief\text{-}summary}$ field perform better than those using all other fields across all label-grouping settings and excel in cases of infrequent labels.

Level Hierarchy. On analyzing the level group results in Figure 6.1, we observe that at level 1, performance is quite similar across fields, including title and $\mathrm{fig\text{-}desc}$. However,

		macro-avg.			micro-avg.		
Model	doc-rep	P	R	F1	P	R	F1
THMM	$\boldsymbol{bs}$	47.8	32.3	36.6	70.4	59.9	64.7
THMM	$\boldsymbol{a}$	39.6	30.0	32.6	66.1	59.8	62.8
THMM	$\boldsymbol{cl}$	41.1	27.5	31.0	67.5	58.0	62.4
THMM	$\boldsymbol{dd}$	42.5	25.1	29.9	66.9	54.1	59.8
THMM	$\boldsymbol{fd}$	38.9	20.1	25.0	65.7	49.3	56.3
THMM	$\boldsymbol{t}$	35.6	19.5	23.6	62.8	49.0	55.0
THMM	$e_f(t + a)$	42.6	**36.7**	37.7	66.6	63.3	64.9
TB	$tfidf(bs)$	46.3	24.5	30.2	68.9	58.9	63.5
TB	$tfidf(bs + a)$	46.8	24.8	30.6	69.5	59.5	64.1
TB	$tfidf(bs + a + cl)$	46.6	24.6	30.1	69.9	60.1	64.6
TB	$tfidf(bs + a + cl + dd)$	45.4	23.4	28.7	69.8	60.8	65.0
TB	$tfidf(bs + a + cl + dd + fd)$	45.5	23.5	28.9	69.8	60.8	65.0
TB	$tfidf(bs + a + cl + dd + fd + t)$	45.5	23.6	28.9	69.9	60.9	65.1
THMM	$\boldsymbol{bs}; \boldsymbol{a}$	45.6	33.3	36.9	68.7	62.1	65.2
THMM	$\boldsymbol{bs}; \boldsymbol{a}; \boldsymbol{cl}$	45.1	33.3	36.6	69.9	61.8	65.6
THMM	$\boldsymbol{bs}; \boldsymbol{a}; \boldsymbol{cl}; \boldsymbol{dd}$	45.4	31.7	35.6	69.8	62.3	65.9
THMM	$\boldsymbol{bs}; \boldsymbol{a}; \boldsymbol{cl}; \boldsymbol{dd}; \boldsymbol{fd}$	45.7	32.3	35.8	69.7	62.3	65.8
THMM	$\boldsymbol{bs}; \boldsymbol{a}; \boldsymbol{cl}; \boldsymbol{dd}; \boldsymbol{fd}; \boldsymbol{t}$	46.6	32.0	36.1	69.9	62.3	65.9
THMM	$\boldsymbol{bs} \oplus \boldsymbol{a}$	46.7	34.8	38.1	69.6	63.1	66.2
THMM	$\boldsymbol{bs} \oplus \boldsymbol{a} \oplus \boldsymbol{cl}$	47.0	34.9	38.3	70.2	63.4	66.6
THMM	$\boldsymbol{bs} \oplus \boldsymbol{a} \oplus \boldsymbol{cl} \oplus \boldsymbol{dd}$	47.5	34.8	38.6	70.6	64.1	67.2
THMM	$\boldsymbol{bs} \oplus \boldsymbol{a} \oplus \boldsymbol{cl} \oplus \boldsymbol{dd} \oplus \boldsymbol{fd}$	48.4	34.7	38.7	70.7	64.4	67.4
THMM	$\boldsymbol{bs} \oplus \boldsymbol{a} \oplus \boldsymbol{cl} \oplus \boldsymbol{dd} \oplus \boldsymbol{fd} \oplus \boldsymbol{t}$	**48.6**	35.0	**38.9**	**70.2**	**65.0**	**67.5**

Table 6.1: Model performance for different document representation inputs. The operators (+), (⊕), and (;) correspond to text concatenation, vector summation, and vector concatenation, respectively, for the USPTO-70k dataset.

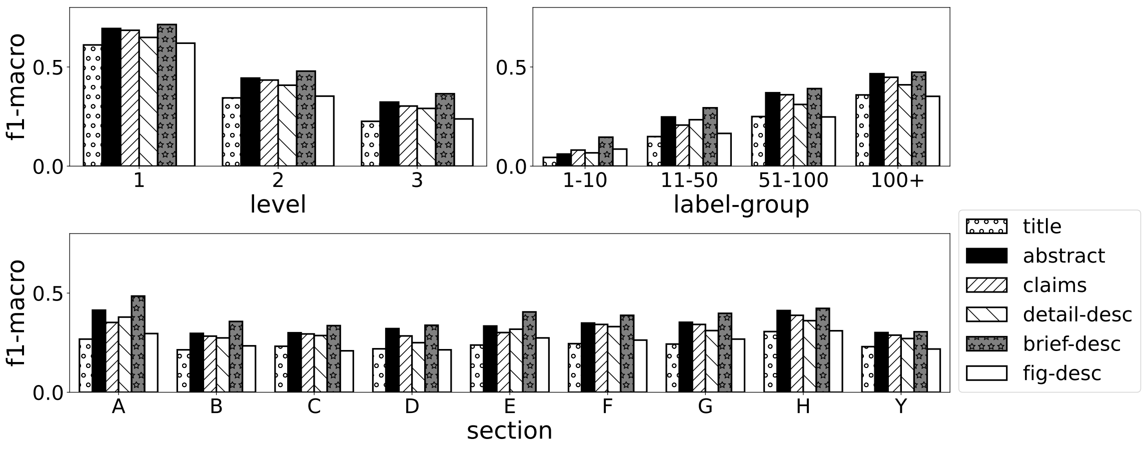


Figure 6.1: Macro-f1 distribution across three label groups that are based on the hierarchical level of the taxonomy (top-left), the instance count for a label in the training set (top-right), and the section label category it belongs (bottom), i.e., labels in the first level of the CPC taxonomy (see Table 4.2, page 60).

			macro-avg.		
model	doc-rep	group	P	R	F1
TB	$tfidf(bs + a + cl + dd + fd + t)$	1-10	1.80	00.7	0.90
TB	$tfidf(bs + a + cl + dd + fd + t)$	11-50	31.7	11.7	16.0
TB	$tfidf(bs + a + cl + dd + fd + t)$	51-100	**63.6**	23.4	32.0
TB	$tfidf(bs + a + cl + dd + fd + t)$	100+	**64.6**	40.8	47.8
THMM	$e_f(t + a)$	1-10	11.1	7.1	7.5
THMM	$e_f(t + a)$	11-50	31.4	25.9	26.2
THMM	$e_f(t + a)$	51-100	45.9	**39.7**	40.9
THMM	$e_f(t + a)$	100+	56.8	**48.5**	51.4
THMM	$\boldsymbol{bs \oplus a \oplus cl \oplus dd \oplus fd \oplus t}$	1-10	**13.1**	**10.7**	**10.8**
THMM	$\boldsymbol{bs \oplus a \oplus cl \oplus dd \oplus fd \oplus t}$	11-50	**43.5**	**27.1**	**31.0**
THMM	$\boldsymbol{bs \oplus a \oplus cl \oplus dd \oplus fd \oplus t}$	51-100	57.4	37.0	**43.2**
THMM	$\boldsymbol{bs \oplus a \oplus cl \oplus dd \oplus fd \oplus t}$	100+	64.2	46.3	**52.5**

Table 6.2: Macro scores for the best-performing model and baselines for different frequency-based groups, which contains labels based on their instance count in the training set of the USPTO-70k dataset (see Section 4.3).

for more fine-grained labels, especially at level 3, the brief-summary is more informative than other field embeddings.

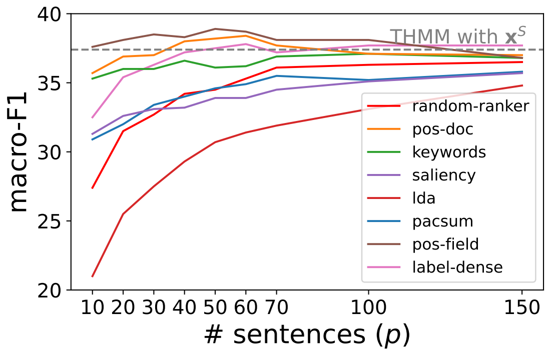
Frequency-based Groups. For the high-frequency label group, we see similar performance for abstract, claims, and brief-summary. However, for labels with fewer instances, the performance for brief-summary is noticeably better.

Section/Domain. Using brief-summary consistently leads to better results across different sections, as does abstract to a lesser degree (Figure 6.1). However, in the case of D, H, and Y as label groups, the relative gain in performance with brief-summary is marginal compared to abstract. Here, D, H, and Y represent the section labels (labels in the first level of the CPC taxonomy) with the descriptions “Textiles; Paper”, “Electricity”, and “General Tagging of New Technological Categories...”, respectively.

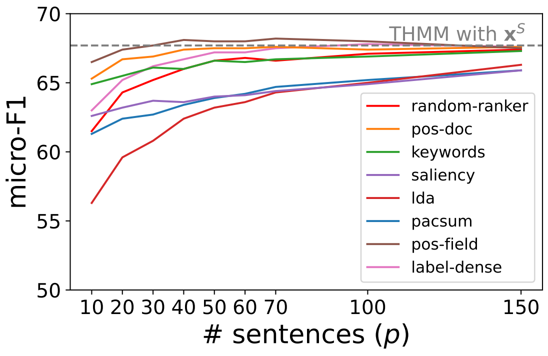
Summary. The experiments identify brief-summary as the most informative patent field and vector summation as the most effective aggregation technique when aggregating field information. Neural models outperform their non-neural counterparts, especially in cases of infrequent labels. Further, we find that optimal performance can be reached by using a subset of the most informative fields for both neural and non-neural classifiers. Conversely, incorporating additional field information leads to a marginal gain or, in some cases, a decrease in classification performance.

6.5.2 Performance for Sentence-Based Representations

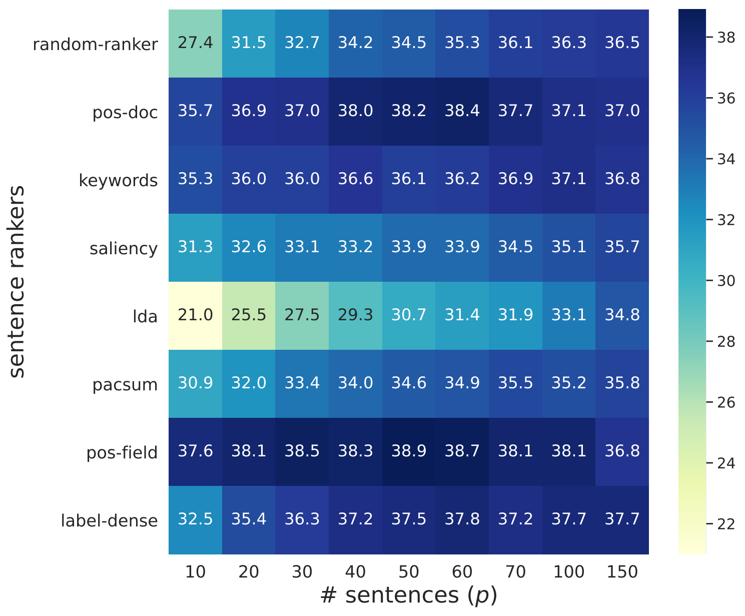
In this section, we evaluate the effectiveness of different sentence rankers when identifying a subset of sentences for the CPC classification task. For a given sentence ranker, we experiment with different values of the parameter p, i.e., the number of selected sentences, to



(a) macro-F1



(b) micro-F1



(c) macro-F1

Figure 6.2: Performance of sentence rankers in the CPC classification task for different values of p on the validation set, using *THMM with* $\boldsymbol{x}^{S^*}$ for USPTO-70k.

determine the optimal p value for patent classification. This evaluation is performed on the validation set, and later, we analyze the relative positions of sentences selected with different sentence ranking methods, thus revealing the sentence selection processes of different sentence rankers. Based on this analysis, we propose a hybrid approach that combines different sentence rankers. Finally, we compare the performance of the best-performing sentence ranker and classifier pair against neural and non-neural baselines on the test set. In the following text, the classification performance of a sentence ranker signifies the classification performance of the texts selected by using the sentence ranker.

Determining the Number of Sentences Sufficient for Optimal Classification Performance. The results of the analysis are shown in Figure 6.2. The horizontal lines in Figure 6.2a and 6.2b show the performance of *THMM with* $\boldsymbol{x}^S$ as a baseline, where $\boldsymbol{x}^S$ is a document representation that is generated by combining all sentence embeddings using the vector mean. With a sentence ranker, the sentence set S^* is determined by selecting p sentences from set S and aggregating the corresponding sentence embeddings to form a document representation $\boldsymbol{x}^{S^*}$. Given the classification model THMM, a set of sentence rankers are compared based on the performance of $\boldsymbol{x}^{S^*}$ for different values of p.

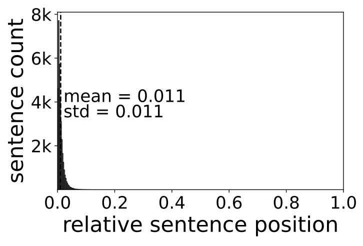
By comparing different sentence rankers to *THMM with* $\boldsymbol{x}^S$ or $\mathrm{random\text{-}ranker}$ as baselines, we find that three sentence rankers, $\mathrm{pos\text{-}field}$, $\mathrm{pos\text{-}doc}$, and $\mathrm{label\text{-}sim}$, perform consistently better than the $\mathrm{random\text{-}ranker}$ in terms of both macro- and micro-F1 scores. Compared to *THMM with* $\boldsymbol{x}^S$, it is found that the performance of these three rankers is equivalent or comparable when $p \geq 40$. In general, pacsum, lda, and $\mathrm{saliency}$ rankers perform poorly compared to $\mathrm{random\text{-}ranker}$. The overall macro-F1 score of the $\mathrm{keyword}$ ranker is better than that of the $\mathrm{random\text{-}ranker}$ ranker, whereas in terms of micro-F1, the $\mathrm{keyword}$ ranker is better for lower p values.

In Figures 6.2a and 6.2b, it can be seen that for higher p values $\mathrm{pos\text{-}field}$, $\mathrm{pos\text{-}doc}$, $\mathrm{label\text{-}sim}$, and $\mathrm{keyword}$ approach the performance achieved when considering all the sentences in the document S. Also, adding more sentences helps improve the performance of the three worst-performing rankers, pacsum, lda, and $\mathrm{saliency}$. The performance of these three rankers shows an upward trend with a gradual increase in p, and their macro-F1 approaches *THMM with* $\boldsymbol{x}^S$. Even the random-ranker approaches high performance comparable to $\mathrm{pos\text{-}field}$ with just 150 sentences, which is much less than the average sentence count for patents (more than 300 sentences per patent). However, it is evident that for lower values of p, an effective sentence ranking method can identify informative sentences.

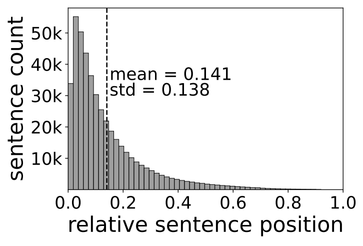
Figure 6.2c shows the numeric values for macro-F1 scores plotted in Figure 6.2a. Looking through Figure 6.2c and focusing on three best-performing rankers, $\mathrm{pos\text{-}field}$, $\mathrm{pos\text{-}doc}$, and $\mathrm{label\text{-}sim}$, it can be observed that optimal performance can be achieved with just 40-60 sentences. Among all the sentence rankers, the $\mathrm{pos\text{-}field}$ ranker performs best with 50 sentences.

Analysis of Relative Sentence Position. A sentence ranker assigns a score to each sentence within a document. When selecting the top-ranked sentences as input to a classifier, it is important to know which part of the document and patent fields the sentences belong to. To achieve this, we determine the position of sentences within the document, then analyze the distribution of sentence positions for different patent fields, and finally analyze the positions of the top-50 sentences selected with a ranker. This analysis reveals that similar classification performance can be achieved with a diverse set of sentence rankers that prefer sentences from varied sentence positions.

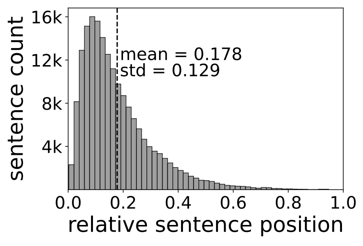
As a first step, we plot the distribution of sentence positions for different patent fields. Patents vary considerably in length; therefore, sentence positions are normalized by dividing a sentence position value by the sentence count for the document to which the sentence belongs. For example, for a sentence s_j, its position in the document is represented by the attribute $s_j.position\_doc$, and the normalized position value is calculated as $s_j.position\_doc/|S|$, which takes a value between 0 and 1 (refer to DIDM defined in



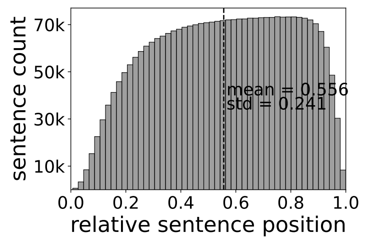
(a) abstract



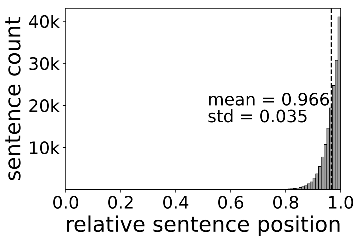
(b) brief-summary



(c) fig-desc



(d) description



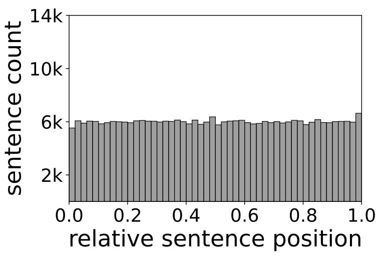
(e) claims

Figure 6.3: Distribution of sentence positions for different patent fields for USPTO-70k.

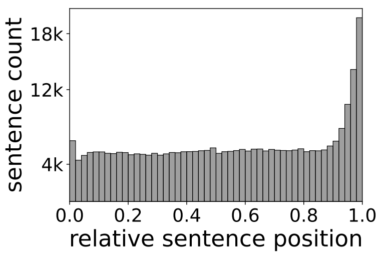
Section 3.2.3). Figure 6.3 shows the distribution of sentence positions. The abstract (Figure 6.3a), brief-summary (Figure 6.3b), and fig-desc (Figure 6.3c) fields are located within the first 40% of the sentences in a document, whereas sentences associated with the claims section appear in later parts of a document, usually within the last 10% of the sentences in the document. The sentences within the detail-desc field take a wide range of position values (Figure 6.3d).

We analyzed performance on the validation set and found that optimal classification performance can be achieved with just 50 sentences. Here, we analyze the relative position of sentences selected with different sentence rankers. As a random-ranker assigns the same score to all sentences within the document and selects the sentences at random, it produces a uniform distribution of sentence positions (Figure 6.4a). For the three worst-performing sentence rankers, pacsum (Figure 6.4e), lda (Figure 6.4d), and saliency (Figure 6.4c), the distribution of sentence positions is similar to that of a random-ranker (Figure 6.4a), in which case sentences in abstract and brief-summary are assigned lower weights.

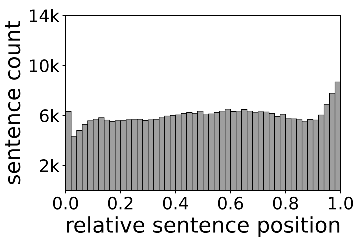
Comparing position-based rankers, we find that pos-doc selects sentences from the initial part of a document (Figure 6.4f), primarily from abstract and brief-summary. In contrast, pos-field (Figure 6.4h) gives preference to sentences appearing earlier in a field and selects sentences across fields. When compared to pos-doc, it is found that the sentences within the claims field are preferred by the pos-field ranker. Similarly, the keywords ranker gives higher priority to the sentences in claims that contain crucial terms defining an invention (Figure 6.4b). By looking into Figure 6.4g, we find that the label-dense sentence ranker picks important sentences from abstract, brief-summary, and claims. Also, it is able to identify sentences in detail-desc, which have a higher semantic similarity with the CPC/IPC label descriptions.



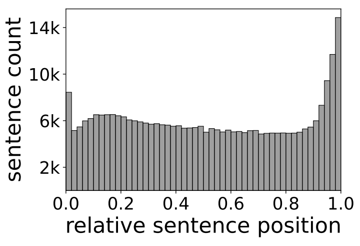
(a) random-ranker



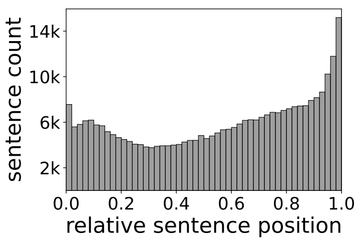
(b) keywords



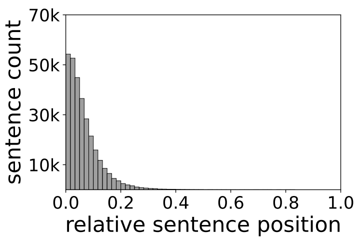
(c) saliency



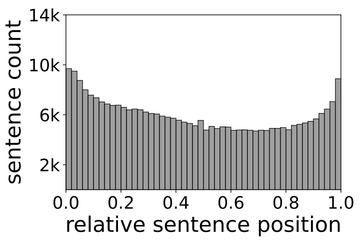
(d) lda



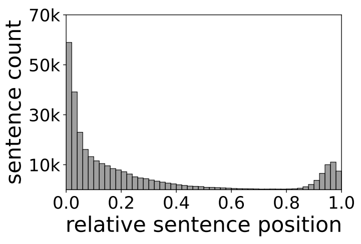
(e) pacsum



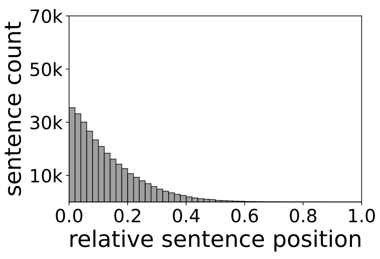
(f) pos-doc



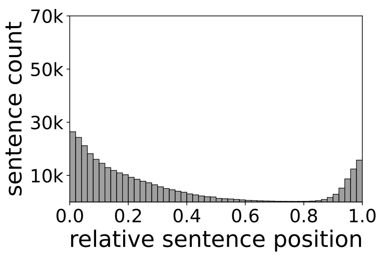
(g) label-sim



(h) pos-field



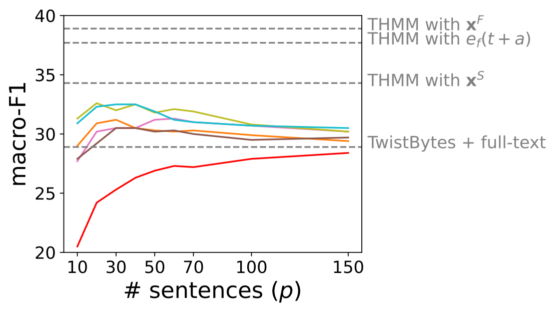
(i) pos-doc_label-sim



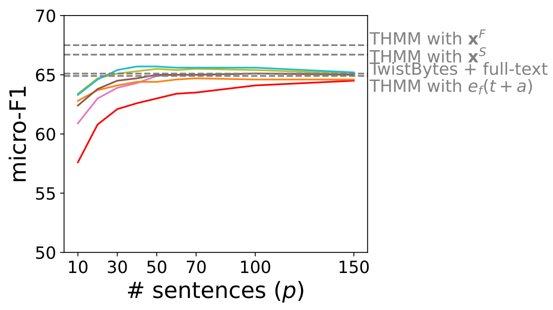
(j) pos-field_label-sim

Figure 6.4: Distribution of sentence positions for top-50 sentences extracted from the USPTO-70k validation set by various sentence rankers.

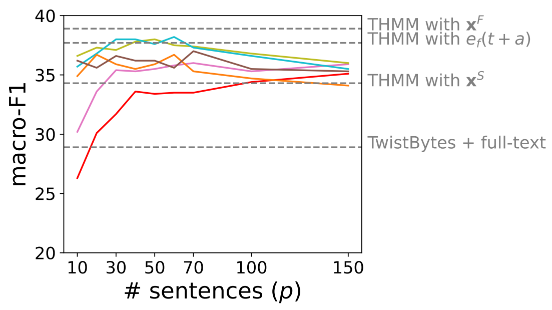
Performance with Hybrid Rankers. Our analysis identifies pos-field, pos-doc, and label-sim as the three best-performing sentence rankers, and later while analyzing sentence positions, we find that each of them selects sentences from different parts of a document (Figure 6.4). Thus, we might achieve better classification performance by combining the scores assigned by these rankers to select informative sentences. We propose two hybrid rankers combining label-sim with two position-based rankers, pos-doc and pos-field, which are referred to as pos-field_label-sim and pos-doc_label-sim, respectively (further details are provided in Section 6.3.2). In this case, a sentence score is computed by giving equal weights to the sentence scores from the two rankers.



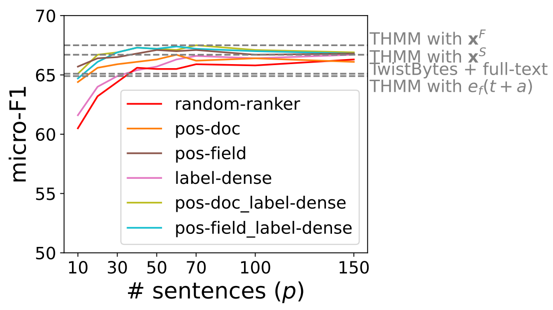
(a) macro-F1 for *TwistBytes with* TF-IDF vector generated from S^*.



(b) micro-F1 for *TwistBytes with* TF-IDF vector generated from S^*.



(c) macro-F1 for *THMM with* $\boldsymbol{x}^{S^*}$

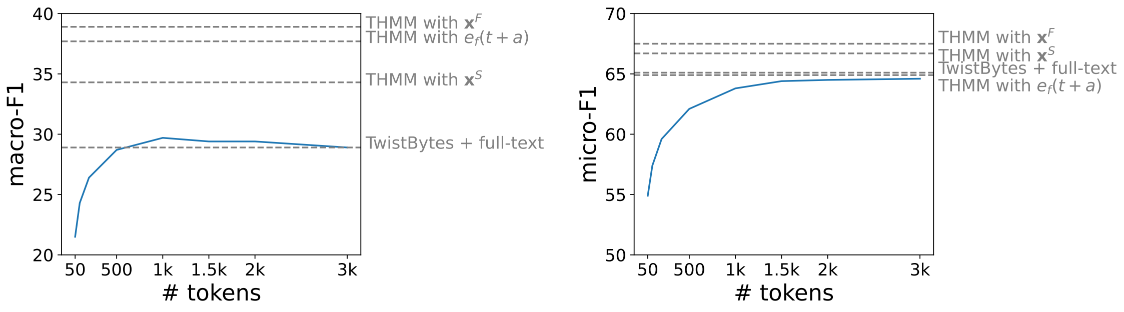


(d) micro-F1 for *THMM with* $\boldsymbol{x}^{S^*}$

Figure 6.5: Performance of sentence rankers with different values of p using neural (see THMM in Section 5.3) and non-neural (TwistBytes based on Benites (2019)) classification models and the USPTO-70k test set. TwistBytes takes a TF-IDF vector as input generated by concatenating sentences from S^*. THMM is used with $\boldsymbol{x}^{S^*}$ computed as a vector mean over embeddings corresponding to sentences in S^*.

We see a small gain in the macro-F1 score when combining sentence scores from multiple top-performing sentence rankers (Figures 6.5a and 6.5c), whereas the micro-F1 score remains almost unchanged (Figures 6.5b and 6.5d). In addition, by comparing sentence positions, we find that if $\mathrm{pos\text{-}doc}$ (Figure 6.4f) scores are enriched using $\mathrm{label\text{-}sim}$ (Figure 6.4g), the sentence extractor selects informative sentences appearing later in the document (Figure 6.4i). Similarly, when looking through Figure 6.4j and comparing sentence positions for $\mathrm{pos\text{-}field\_label\text{-}sim}$ to those for $\mathrm{pos\text{-}field}$ (Figure 6.4h), we find that $\mathrm{pos\text{-}field\_label\text{-}sim}$ picks informative sentences from the later parts of the $\mathrm{abstract}$, $\mathrm{brief\text{-}summary}$, and claims fields.

Comparison with Baselines. We select the best-performing ranker, i.e., $\mathrm{pos\text{-}field\_label\text{-}sim}$, and compare the performance of neural (*THMM with* $\boldsymbol{x}^{S^*}$) and non-neural (TwistBytes with TF-IDF vector generated from S^*) classifiers to the baselines for different values of the parameter p. When comparing micro-F1 scores, the sentence-based neural classifier (THMM with $\boldsymbol{x}^{S^*}$) performs consistently better than THMM with $e_f(t+a)$ for all values of $p \geq 20$ (Figure 6.5d), and exhibits comparable performance in terms of the macro-F1



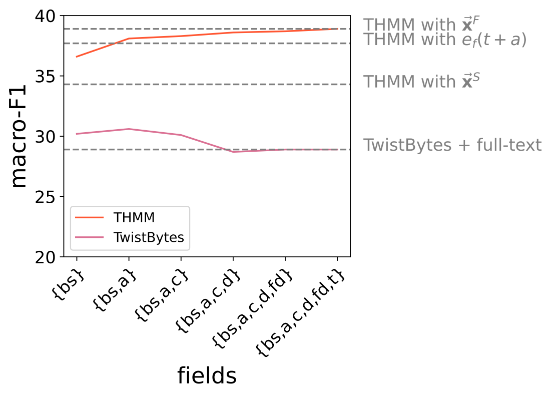
(a) macro-F1 for TwistBytes. **(b)** micro-F1 for TwistBytes.

Figure 6.6: Classification performance for top-k tokens with TwistBytes (Benites, 2019) and $tfidf(.)$ for the USPTO-70k test set.

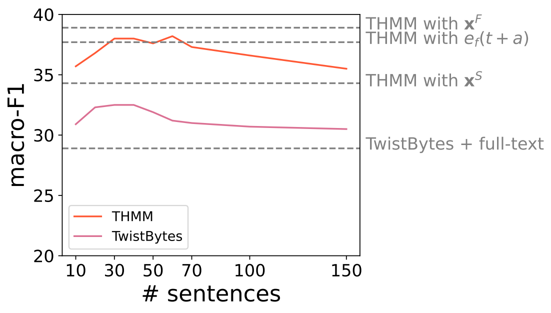
score for $50 \leq p \leq 70$ (Figure 6.5c). Comparing the non-neural classifiers, we find that the sentence-based non-neural classifier performs consistently better than TwistBytes with the full texts of patents as a baseline for macro-F1 and shows comparable performance when evaluated for micro-F1 with $p \geq 30$ (Figures 6.5a and 6.5b). Although the performance of $\mathrm{pos\text{-}field\_label\text{-}sim}$ is not better than the best-performing model from this chapter (*THMM with* $\boldsymbol{t} \oplus \boldsymbol{a} \oplus \boldsymbol{cl} \oplus \boldsymbol{dd} \oplus \boldsymbol{bs} \oplus \boldsymbol{fd}$), still, we demonstrate the potential of extractive summarization techniques in selecting informative texts for neural representation in the context of patent classification.

6.5.3 Performance for Token-Based Representations

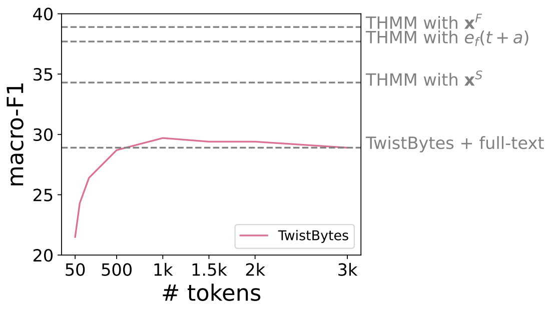
As discussed in Section 6.3.3, a set of key phrases is extracted, arranged in order of importance, and concatenated. The concatenated text is tokenized using a white-space tokenizer, and k tokens are selected to generate a TF-IDF vector. In the current section, we perform an analysis by varying the value of parameter k and determining the performance of a non-neural classifier, TwistBytes with $tfidf(.)$ (see Section 2.4.1), in the CPC classification task. As shown in Figure 6.6a, the macro-F1 score increases with an increase in k, reaches the maximum value for 1k tokens, and gradually converges to the performance of TwistBytes with the full texts of patents. On the other hand, the micro-F1 score increases gradually and tends to show optimal classification performance for a token count value of 1.5k or more (Figure 6.6b). For both macro- and micro-F1 scores, we can conclude that by determining the most important tokens in a document using TextRank, optimal classification performance can be reached using a subset of tokens rather than considering the full texts of patents.



(a) macro-F1 for different sets of fields.



(b) macro-F1 for different values of p, i.e., number of selected sentences, for pos-field_label-sim ranker.



(c) macro-F1 for different values k, i.e., number of selected tokens.

Figure 6.7: Distribution of macro-F1 scores on aggregating information at three levels of granularity (field, sentence, and token).

6.6 Conclusions

One of the pertinent questions in the context of patent classification is whether we need the complete texts of patents or whether we can achieve optimal classification performance with only a subset of text elements. Taking a data-driven approach, we answer this question by evaluating document representation techniques that consider a subset of the most informative fields, sentences, and tokens. Our analysis demonstrates that a subset of the highest-ranked semantic elements can be sufficient to achieve optimal classification performance with an effective ranking mechanism (Figure 6.7). We propose a novel sentence ranker, pos-field_label-dense, that performs comparably to the top sentence rankers associated with baseline extractive summarization techniques. Further, we show that the extractive summarization techniques are effective in the task of selecting informative texts for generating neural representations in the context of patent classification. Furthermore, the best performance on the CPC classification task is attained by aggregating the field embeddings using vector summation. The proposed patent representation is further enriched with CPC/IPC labels in the next chapter and evaluated on the PLS classification task.

Chapter 7

Neural Representations for Patent Landscape Study

In contrast to the CPC/IPC classification task (see Definition 2.1, page 18), in the PLS classification task, semantically rich CPC/IPC labels are associated with a patent, and these can be exploited to improve classification accuracy. Moreover, the evaluations in Chapter 6 demonstrated that a document representation generated by taking a vector sum of field embeddings is the best approach for CPC classification. Building on the findings of Chapter 6, in this chapter, we explore methods that combine patent text and CPC/IPC labels for generating patent representations for the PLS classification task. We demonstrate that the proposed neural computation model performs robustly across PLS datasets.

In Section 7.1, we discuss motivations for document representation techniques that incorporate CPC label information and patent text in the context of the PLS classification task. In Section 7.2, we highlight the limitations of prior work on PLS, based on which we propose novel neural computation models in Section 7.3. We describe the experimental setup in Section 7.4 and evaluate the proposed techniques on the PLS classification task with the PLS datasets (see Section 4.4) in Section 7.5. Additionally, we evaluate a similar technique for a multi-label classification task—classifying PubMed articles into a set of categories relating to COVID-19 (Chen et al., 2021b)—to demonstrate the applicability of the proposed approach to documents other than patents (Section 7.6).

7.1 Motivation and Contributions

Patent Landscape Studies (PLS) are critical business applications that provide crucial insights into the patent landscape for a particular invention or field, enabling organizations, policymakers, and investors to make informed decisions (Cockburn and Macgarvie, 2009). A PLS typically involves three steps: patent search, categorization, and analysis. During the patent search step, a domain expert uses proprietary or open-source search systems, often with keywords and CPC labels as search queries, to identify patents within the scope of the PLS. This is followed by the categorization step, in which a domain expert manually or semi-automatically assigns patents to PLS-oriented categories. Finally, the labeled dataset is analyzed to generate a PLS report, which provides insights into key players and their patent portfolios (see Section 2.2).

Previous works on automating the PLS process have primarily focused on the first step, i.e., the identification of relevant documents (Abood and Feltenberger, 2018; Giczy et al., 2022; Choi et al., 2022). In contrast, in this chapter, we focus on the patent categorization step, which involves assigning a set of user-defined PLS-oriented categories to a patent that is already labeled with a set of CPC/IPC labels. Moreover, the techniques developed for PLS classification may be appropriate for application in other use cases, such as patent alert generation systems (Richter and MacFarlane, 2005). The PatBase system provides functionality to register for periodic patent alerts using keywords and CPC/IPC labels, which may result in a large number of false positives.[1] A patent relevance classification similar to classification in the PLS context might help to reduce the number of false positives produced by the patent alert system. In Section 2.2, we discussed the PLS use case and the associated patent classification task, referred to as the PLS classification task (see Definition 2.2, page 19).

In Chapters 5 and 6, we explored techniques for patent classification that target the CPC classification problem using only the patent text. In Chapter 5, we introduced a transformer-based multi-task model (TMM) and its hierarchical variant (THMM), both of which were evaluated on the CPC classification task using the patent title and abstract as input (see Section 5.3). Additionally, our analysis in Section 4.6 revealed that titles and abstracts are often duplicated across patents, which motivated the document representation techniques proposed in Chapter 6. In Chapter 6, we found that a classifier applying the THMM model performs best when provided with a document representation generated by summing all field embeddings (see Section 6.5.1).

The document representation techniques introduced in the previous chapters did not consider metadata. This was because metadata can introduce bias, be ambiguous, or be only partially known in the context of CPC classification. For example, the inventors and assignee names associated with a patent may introduce bias and are often ambiguous (Haak et al., 2012). Additionally, crucial citations are typically unknown during the filing of the application and later added by the patent examiner (Alcácer et al., 2009). However, in the case of PLS classification, the CPC labels for a document are known and can be incorporated into a document representation (see Section 2.2). By employing the label descriptions in the label_dense sentence ranker and evaluating the top-selected sentences on the classification task (see Section 6.5.2), we can conclude that they comprise crucial domain information, which can be further utilized for the PLS classification task.

In Chapter 6, we found that a document representation generated by summing up all field embeddings achieves the best performance. Inspired by this finding and building on

[1]https://minesoft.com/minesoft-alerts/ [last accessed October 12, 2023]

the work of Choi et al. (2022), we conduct experiments with a combination of text- and label-based feature vectors. We start by exploring different techniques for representing CPC labels and subsequently evaluate the representation generated by combining the best-performing CPC embeddings with the patent-text embedding.

We generate patent text embeddings using SciBERT (Beltagy et al., 2019) for the patents' title and abstract, claims, and description. Additionally, we compare different approaches for generating CPC/IPC label embeddings based on a label co-occurrence graph and by using texts corresponding to label descriptions of CPC labels (see Section 2.1.3). A label co-occurrence graph is generated by exploiting the implicit co-occurrence relation between the CPC/IPC labels. A co-occurrence graph consists of CPC/IPC labels as nodes, and two nodes are connected with an undirected edge if the corresponding labels co-occur in a patent. Next, we apply node2vec on a CPC/IPC co-occurrence graph to generate graph-based label embeddings (see Section 2.5). Further, a label description is associated with each CPC/IPC label that can be used to generate embedding or TF-IDF feature vector (see Figure 2.4, page 16). In Section 7.3, we describe different methods of generating CPC/IPC label embeddings that are used in our experiments. The representations for a patent text and corresponding CPC/IPC labels are combined using vector concatenation.

We evaluate the proposed techniques against neural and non-neural baselines, and the evaluation results are discussed in Section 7.5. Our experiments show that the TF-IDF-based representation of the label description outperforms the neural text- and graph-based representations. Furthermore, among the dense representation methods, the graph-based representation is superior to the text-based representation. Similar to the findings in Section 6.5.1, we observe that using all textual fields improves performance. Finally, we observe that our approach of combining patent text embeddings with graph-based CPC/IPC embeddings leads to a robust method that consistently outperforms all baselines across three PLS datasets. Our analysis of the performance of classifiers trained with varying training instance counts also demonstrates that 200-300 instances are sufficient to train a PLS classifier.

To further highlight the significance of metadata such as pre-assigned labels in document classification tasks, we conduct an out-of-domain evaluation on a related task: labeling research publications using categories relating to COVID-19, i.e., Task 5 of the BioCreative VII challenge (Chen et al., 2021b). Each publication consists of title and abstract as text fields, and pub_type, journal, and keywords as metadata. By comparing different metadata, we find that the representation based on keywords performs best when used with TMM. Like the best-performing approach for PLS classification, the best performance is achieved by generating a document representation that combines keywords with the publication text. Examining the statistics provided by the task organizers, we observe that the proposed classifier outperforms the baseline by 13% in relative terms and is superior to 75% of submissions made during the challenge (Section 7.6).

Contributions. In summary, the main contributions of this chapter are as follows:

- We propose a **neural computation model** for the PLS classification task that exhibits **robust performance** across all three PLS datasets.
- We demonstrate the applicability of our proposed approach to an out-of-domain evaluation: classifying PubMed into COVID-19-related categories. This demonstrates that classification techniques primarily developed for patents can also be applied outside of patents in a similar task setting.

7.2 Related Work

Prior works on classification models were discussed in Section 5.2, whereas Section 6.2 discussed previous works on document representation methods in general and patent representation in particular. This chapter proposes classification techniques in the context of a PLS. These techniques generate a patent representation by incorporating CPC label information and patent text. Below, we discuss the key related works in this context.

PLS has applications in a variety of fields, such as gene therapy (Picanço-Castro et al., 2020), blockchain (Clarke et al., 2020), research on advances in CO_2 utilization technologies (Norhasyima and Mahlia, 2018), and analysis of biodiesel production techniques (Mahlia et al., 2020). The number of patents included may vary according to the scope of the PLS. For example, Dumet et al. (2019) restricted their study to 37 patent families, whereas Clarke et al. (2020) included thousands of blockchain-related patents.

We are aware of several works which aim to automate the first step of the PLS process. Lee and Lee (2019) propose a methodology for performing a PLS that uses a vector space model to identify relevant novel inventions. Abood and Feltenberger (2018) first identify relevant patents by expanding a seed list, performing forward and backward traversal on the patent citation graph, and restricting the candidate patents by using relevant CPC labels. They then train a classifier using one-hot embeddings of references and CPC codes and an embedding of the patent abstract using word2vec and an LSTM to predict whether a patent is relevant to a PLS. Similarly, for a PLS in the artificial intelligence domain, Giczy et al. (2022) propose a classifier that concatenates the LSTM output for abstracts and claims to the citation embedding. Choi et al. (2022) employ a model architecture that is more similar to our own, using a transformer to embed abstracts and the graph neural network diff2vec (Rozemberczki and Sarkar, 2018) to embed CPC labels. In contrast to these works, we target the second step of the PLS process, categorizing a set of retrieved documents into business- or application-oriented categories.

In the classification (Richter and MacFarlane, 2005; Benites et al., 2018) and clustering (Vlase et al., 2012) contexts, non-neural count- and TF-IDF-based feature vectors reflecting IPC, inventor, and assignee information have been proposed. For CPC classification, Niu and Cai (2019) index label descriptions, retrieve them using the input text as a query, and generate an input representation concatenating a vector containing BM25 scores corresponding to the label descriptions to the BERT-based sequence representation. Fang et al. (2021) compute embeddings on the basis of graphs constructed from word co-occurrence, inventor, and assignee information and combine them using attention-based sums. In contrast, we decided to restrict our study to content-based features, as using inventor and assignee information might introduce biases that contradict the goal of a PLS. For example, an organization that is predominantly involved in mobile technologies ventures into automobile research. In that case, the classifier will have an inherent bias toward labels related to mobile technologies labels when using the organization's name as input, irrespective of the patent's content.

Another related research direction is that of using label embeddings for the target label to generate label-specific representation (Mullenbach et al., 2018; Dong et al., 2020; Cai et al., 2020; Liu et al., 2021). This is usually done by computing attention weights between label embeddings and embeddings corresponding to the sentences or tokens in the input sequence. Next, the attention weights are used to combine embeddings corresponding to sentences or tokens to generate label-specific representations, which are then used for predicting labels individually. The methods proposed by related work mainly differ in their choices of label and text representation techniques. The input text has been represented using CNNs (Mullenbach et al., 2018), BiLSTMs (Xiao et al., 2019; Vu et al., 2020), and BERT (Dong et al., 2020; Cai et al., 2020; Liu et al., 2021). Due to the effectiveness of BERT for representing text semantics, some recent work uses it to represent label descriptions (Liu et al., 2019; Dong et al., 2020; Xiong et al., 2021). In these studies, the label descriptions are provided as input to BERT and generate label embeddings, which are then used to weigh the tokens to generate label-specific input representations. Also, Cai et al. (2020) leverage the hierarchical taxonomy structure and use a graph convolutional network (Kipf and Welling, 2017) to represent labels. A few of these methods use randomly initialized parameters as label embeddings, which are learned jointly with the model parameters during training (Mullenbach et al., 2018; Xiao et al., 2019; Vu et al., 2020). In contrast to the related work described above, in the content of PLS, the CPC/IPC labels provide informative descriptions and are known during inference. Therefore, we can directly incorporate them into the document representation.

7.3 Methodology

Our evaluation in the previous chapter demonstrates that the best performance is achieved by the document representation combining all field embeddings using vector summation. In the PLS classification task context, the CPC/IPC labels are known and can be incorporated into a document representation. This section introduces computational models that predict PLS-oriented categories by generating a document representation that combines the text of patents with their associated CPC labels, as shown in Figure 7.1.

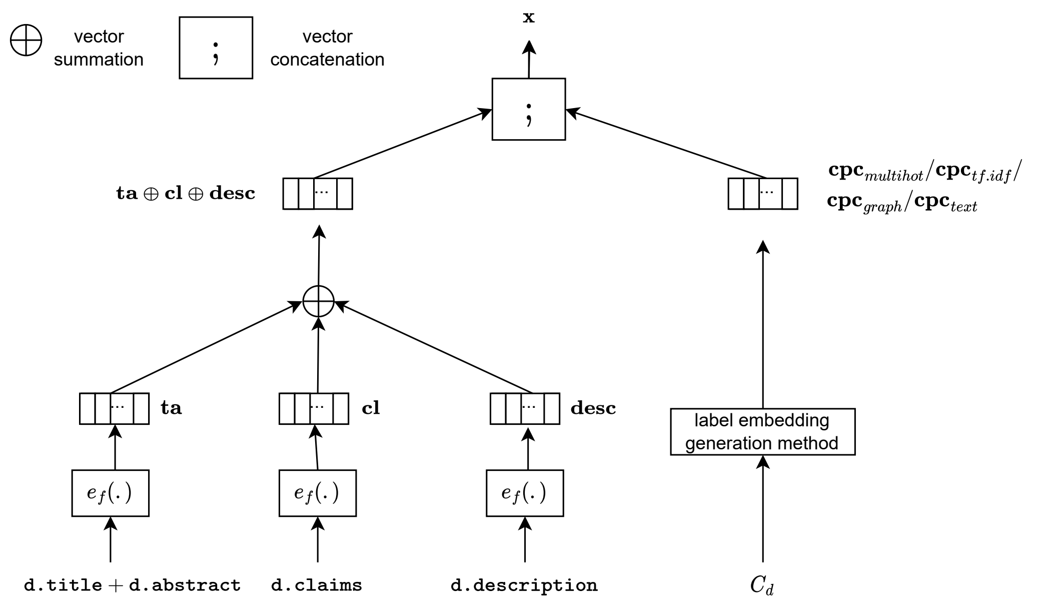

vector summation
;
vector concatenation
x
ta ⊕ cl ⊕ desc
$\mathbf{cpc}_{multihot}/\mathbf{cpc}_{tf.idf}/$ $\mathbf{cpc}_{graph}/\mathbf{cpc}_{text}$
ta
cl
desc
label embedding generation method
$e_f(.)$
d.title + d.abstract
d.claims
d.description
C_d

Figure 7.1: Generating a patent representation $\boldsymbol{x}$ for the PLS classification task.

To understand the classification process, we refer to Section 3.2, which describes the specifics of the classification pipeline. As a first step, an instance of the patent document model is mapped to an instance of the domain-independent document model, which is then mapped to an instance of DocRepDM specific to the document representation method. The proposed patent representation method takes an instance of DocRepDM_Chunks_Label as input, which is comprised of field texts as chunks and CPC/IPC labels as source_labels (see Section 3.2.4). Each Label_DocRepDM instance in source_labels contains the textual description of a CPC/IPC label as text attribute and associated embedding as ml_feature attribute. Next, we define the methodology for generating a patent representation from an instance of DocRepDM_Chunks_Label.

Neural Text Representations. Following the process similar to the field-based representation proposed in Section 6.3, we use SciBERT model and generate three text embeddings: $e_f(t+a)$ using the concatenated text of the title and abstract, $e_f(cl)$ using the claims text, and $e_f(desc)$ using the text of the description text. The method $e_f(\cdot)$ represents the process for generating a sequence embedding using a transformer-based language model, fine-tuned for the CPC classification task (see Section 5.5). For brevity, we refer to the embeddings $e_f(t+a)$, $e_f(cl)$, and $e_f(desc)$ as $\boldsymbol{ta}$, $\boldsymbol{cl}$, and $\boldsymbol{desc}$, respectively. Based on the finding of Section 6.5, we generate a patent text representation by combining the field embeddings using a summation operation ($\oplus$).

CPC Label Embeddings. As discussed above, each of the source_label instances has text and ml_feature attributes. The associated information about CP/IPC labels can be used to generate a patent representation. We experiment with four different ways of em-

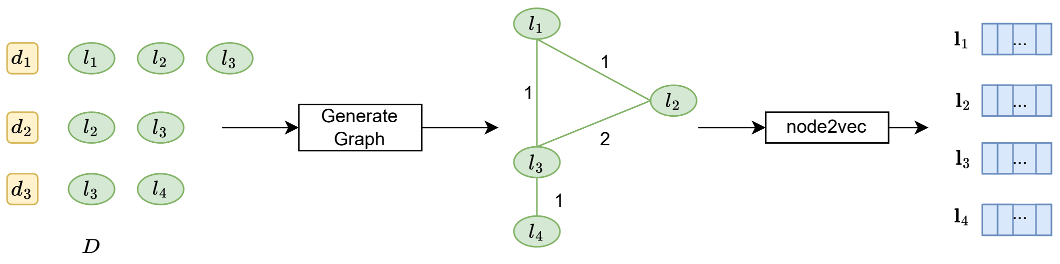


Figure 7.2: A co-occurrence graph is generated based on the co-occurrence relation between CPC/IPC labels in the document set D, which is provided as input to the graph-based representation method, i.e., node2vec, to generate label embeddings.

bedding knowledge about the CPC labels associated with a patent document, which are described as follows:

- $\boldsymbol{cpc}_{multihot}$: The simplest embedding consists of a multi-hot encoded vector with each dimension indicating the presence of one CPC label, referred to as $\boldsymbol{cpc}_{multihot}$.

- $\boldsymbol{cpc}_{text}$: The label description associated with a CPC/IPC label represents an important domain concept (see Figure 2.4, page 16). It contains crucial semantic information, which can be leveraged to enrich a patent representation. In this case, a label description is passed through SciBERT, and the representation corresponding to the last hidden state of the [CLS] token is considered a label embedding. The SciBERT model is fine-tuned for the CPC classification task with concatenated text from title and $\mathrm{abstract}$ as input (see Section 5.5). As discussed above, $\mathrm{source\_labels}$ is a list of $\mathrm{Label\_DocRepDM}$ instances, which, in the current setting, represent the CPC/IPC labels in a patent. The $\mathrm{ml\_feature}$ attribute in a $\mathrm{Label\_DocRepDM}$ instance represents the corresponding CPC/IPC label embedding. A 768-dimensional representation is generated by averaging all the label embeddings corresponding to the CPC labels associated with a patent, which we refer to as $\boldsymbol{cpc}_{text}$.

- $\boldsymbol{cpc}_{tf.idf}$: The CPC label descriptions contain important domain-specific keywords, and this information can be captured with TF-IDF vectors. As the first step, the term statistics are computed with descriptions corresponding to all the subgroup labels (see Figure 2.4, page 16). The $\mathrm{source\_labels}$ attribute is a list of $\mathrm{Label\_DocRepDM}$ instances, representing CPC/IPC labels in a patent, and the text attribute in $\mathrm{Label\_DocRepDM}$ represents the corresponding label description. The label descriptions corresponding to the CPC labels associated with a patent are concatenated and provided as input to the TF-IDF model, thus generating a 140k-dimensional TF-IDF feature vector referred to as $\boldsymbol{cpc}_{tf.idf}$.

- $\boldsymbol{cpc}_{graph}$: When representing the CPC/IPC labels as a co-occurrence graph, we assume that the frequently co-occurring labels have higher proximity in semantic space than the less frequent ones. The process of creating a co-occurrence graph and generating graph embeddings is depicted in Figure 7.2, whereas the specifics of the node2vec algorithm are discussed in Section 2.5. In a co-occurrence graph, the nodes correspond to the labels in the CPC/IPC taxonomy. A pair of nodes are connected if the corresponding labels occur within a patent. Further, an edge weight indicates the number of such co-occurrences. The co-occurrence graph is provided as input to a node2vec algorithm, which generates an embedding for each node and, thus, for each CPC/IPC label (see Section 2.5). Similar to $\boldsymbol{cpc}_{text}$, a document-level representation is generated by taking the mean of all the label embeddings corresponding to the CPC labels associated with a patent, which is referred to as $\boldsymbol{cpc}_{graph}$.

Classification Model. We use the Transformer-based Multi-task Model (TMM) described in Section 5.3 as the classification model in the proposed architecture. The model takes as input either the CPC-label embeddings, the text-based embeddings, or a combination of both obtained through vector concatenation (;). The TMM model uses one classification head for each label, and each head contains three dense layers. The final dense layer for each head has a binary softmax output that predicts whether or not a label applies to an input document.

7.4 Experimental Setup

This section describes the experimental setup used to evaluate various classifiers on the PLS classification task. We provide details about the datasets, evaluation metrics, hyperparameters, and baselines used in our experiments.

Datasets. We evaluate our PLS classifiers on three datasets from two diverse domains, biochemical and mechanical, as described in Section 4.4. The two World Intellectual Property Organization (WIPO) datasets, Atz and Rito, and the InjVal dataset exhibit different characteristics in terms of scope, patent offices, publication years, patent language, and number of labels, as discussed in Section 4.4. The target labels in the WIPO datasets have an average count well above one. In contrast, in InjVal, the average count is approximately equal to one. This difference indicates multi-label and single-label classification settings, respectively.

For evaluation, each dataset is split into two parts. The heldout set contains 15% of the total instances and is never seen during model fine-tuning and development. The remaining 85% of the data are used for 5-fold cross-validation. In each fold, three folds are used for training, one for tuning the models, and one as the validation set. Finally, each model is evaluated on the heldout set, and the mean and standard deviation values are reported for these five evaluations.

Evaluation Metrics. Unlike CPC/IPC classification, PLS classification for the three PLS datasets is a flat classification problem (Silla and Freitas, 2011). In a flat classification setting, macro- and micro-average scores are calculated for precision, recall, and F1-score (see Section 2.7). The micro-average score is computed over all the instances in the test set and provides an overall estimate of a model's performance. The macro-average score is an average of the performance over all the labels, thus evaluating a model's ability to perform better in a class-imbalance scenario.

Hyperparameters. We define the hyperparameters for the classification model (TMM) and graph embedding algorithm below.

TMM. We iteratively optimize the learning rate, hidden layer, and batch size and arrive at the following hyperparameter values. With optimization, we find optimal corpus-specific learning rates of 1e-5, 3e-5, and 5e-5 for InjVal, Rito, and Atz, respectively. Also, it is found that the hidden layer size of 50 for all dense layers in the classification heads and a batch size of 4 work well across datasets. We use a dropout rate of 0.25, the same as for the CPC classification task in Section 5.4.3. The models are trained for a maximum of 50 epochs with early stopping if the macro-F1 for the validation set does not increase for seven epochs.

Node2Vec. By merging the CPC/IPC co-occurrence graphs for the three datasets, we produce a unified co-occurrence graph that consists of 9.5k nodes and 500k edges. This subgraph is provided as input to the node2vec algorithm, generating label embeddings of dimension 128. The p and q parameters determine whether the next hop is selected from neighboring or non-neighboring nodes (see Section 2.5). We set the p and q values to 1, giving equal weightage to both these cases. For computational efficiency, we perform 10 random walks with a maximum length of 50.

Baselines. We compare our approach against the following state-of-the-art neural and non-neural classifiers:

TMM with $\boldsymbol{ta}$. As a neural baseline, we use the setup proposed in Chapter 5 for multi-label classification with TMM as the classification model. A document representation is generated by concatenating the title and $\mathrm{abstract}$ and providing the result as input to a SciBERT model, represented as $e_f(t + a)$ or $\boldsymbol{ta}$.

SVM. Benites et al. (2018) achieved competitive results with an SVM (see Section 2.6.1) ensemble-based approach in the context of the ALTA 2018 shared task on multi-label IPC classification (Mollá and Seneviratne, 2018). The 140k-dimensional feature vector for the complete document text, i.e., $t + a + cl + desc$, comprises TF-IDF values for 70k character n-grams (3- to 6-chars) and 70k word n-grams (1- to 2-grams).

Dataset	Label	Key phrases
InjVal	Exhaust Line Injector	exhaust line injector ; line injector
InjVal	Bi-Fuel-Injector	bi-fuel-injector ; bi-fuel injector
InjVal	Water Injection	water injection
InjVal	Piezoelectric Actuator Spring	piezoelectric actuator spring
InjVal	Fuel Rail	fuel rail
InjVal	Dual Injection	dual injection
InjVal	Direct Injector Piezo	direct injector piezo
InjVal	Port Fuel Injector	port fuel injector
InjVal	Direct Injector Solenoid	direct injector solenoid
InjVal	Air Injection Valve	air injection valve
InjVal	Piezoelectric Actuators for Injectors	piezoelectric actuators for Injectors ; piezoelectric actuator
InjVal	Pump Injector Combination	pump injector combination ; pump injector
InjVal	Other Injectors (SEV)	other injector
InjVal	Piezoelectric Ceramic Material	piezoelectric ceramic material ; piezoelectric ; ceramic
InjVal	Natural Gas Injector (NGI)	natural gas injector ; gas injector
InjVal	High Pressure Pipe	high pressure pipe ; high pressure
Rito	Pharmaceutical Compositions	pharmaceutical compositions ; composition ; pharmaceutical
Rito	Synthesis and Crystalline Forms	synthesis and crystalline forms ; crystalline form ; synthesis
Rito	Stabilized Forms	stabilized form
Rito	Methods of Treating HIV	methods of treating hiv ; hiv
Rito	Prodrug	prodrug
Rito	Derivatives	derivatives
Rito	Combinations	combination
Atz	Autoimmune - Inflammatory	autoimmune ; inflamatory ; autoimmune - inflammatory ; autoimmune-inflammatory ; autoimmune inflammatory
Atz	Cancer	cancer
Atz	Kaposi	kaposi
Atz	Neurologic	neurologic
Atz	IBD	ibd ; inflammatory ; bowel disease ; inflammatory bowel disease
Atz	Herpes	herpes
Atz	Hepatitis C Virus	hepatitis ; hepatitis c virus ; c virus
Atz	Serine Protease Inhibitor	serine protease inhibitor ; protease inhibitor ; inhibitor

Table 7.1: PLS-oriented categories and associated key phrases.

Runtime. Each model is trained on a single Nvidia Tesla V100 GPU, with early stopping if the macro-F1 score on the validation set does not improve for five epochs. The transformer-based models take 20 to 24 hours to train for the WIPO datasets, whereas the InjVal dataset takes 70 to 80 hours. In contrast, the models which only use metadata information can be trained in one to two hours.

7.5 Results

In Section 7.3, we introduced various content and label embedding methods for representing patent text and CPC labels. Here, we evaluate the patent representations on the PLS classification task to identify a patent document representation that works robustly across PLS datasets. We analyze the performance of different embeddings when considered in-

		macro-avg.			micro-avg.		
Dataset	**Model**	**P**	**R**	**F1**	**P**	**R**	**F1**
InjVal	search with key phrases on full-text	20.85	23.31	16.41	31.03	24.38	27.30
InjVal	search with label name on full-text	17.70	7.52	9.30	51.41	6.32	11.26
InjVal	search with label name on title + abstract	14.94	4.04	6.01	71.43	3.12	5.99
InjVal	our best (TMM with $\boldsymbol{ta} \oplus \boldsymbol{cl} \oplus \boldsymbol{desc}; \boldsymbol{cpc}_{graph}$)	**74.30**	**66.40**	**67.70**	**84.20**	**84.40**	**84.30**
Rito	search with key phrases on full-text	25.67	76.03	34.38	27.48	93.63	42.49
Rito	search with label name on full-text	23.39	42.24	28.24	36.52	68.15	47.56
Rito	search with label name on title + abstract	18.68	5.88	7.06	20.37	7.01	10.43
Rito	our best (TMM with $\boldsymbol{ta} \oplus \boldsymbol{cl} \oplus \boldsymbol{desc}; \boldsymbol{cpc}_{graph}$)	**64.40**	**49.50**	**53.90**	**70.70**	**65.10**	**67.70**
Atz	search with key phrases on full-text	49.11	86.51	56.71	42.53	83.58	56.38
Atz	search with label name on full-text	51.24	60.88	51.27	59.71	61.19	60.44
Atz	search with label name on title + abstract	55.21	11.05	17.11	89.29	12.44	21.83
Atz	our best (TMM with $\boldsymbol{ta} \oplus \boldsymbol{cl} \oplus \boldsymbol{desc}; \boldsymbol{cpc}_{graph}$)	**72.20**	**63.26**	**66.20**	**75.64**	**70.90**	**73.20**

Table 7.2: Comparing performance using a simple baseline of searching label names or associated key phrases in the document text vs. our more sophisticated robust approach.

dividually or in combination and compare them to the neural and non-neural baselines. Finally, we perform an ablation study to determine the minimum number of training instances required to train a PLS classifier (Section 7.5.4).

7.5.1 Baseline with PLS-Oriented Categories

As the exact details of the manual categorization process for the WIPO datasets are unknown, we experimented with a simple baseline that searches for PLS-oriented category names or associated key phrases in the document text. Table 7.1 shows PLS-oriented categories and associated key phrases, while the results of the simple baseline are shown in Table 7.2.

Although the simple baseline results in a high recall for Atz and a relatively high recall for Rito, precision is rather low, highlighting the need for a more robust PLS classification method. For InjVal, an extremely low recall indicates a document creation process with rare or no use of category names or associated key phrases.

Based on our analysis, we conclude that for Atz, the domain expert likely extensively used the label name or associated key phrases to determine a document's relevance to a label. However, despite being supported by NLP technology, as reported in the patent landscape report[2], we assume that the patent professional applied their domain expertise when labeling documents. For the InjVal dataset, we know that the domain expert primarily used IPC codes and associated figures within a patent document to determine its relevance.

[2]https://www.wipo.int/edocs/pubdocs/en/patents/946/wipo_pub_946_2.pdf [last accessed December 10, 2023]

7.5.2 Comparison of Patent Embeddings

We assess the efficacy of CPC-label and text embeddings on the PLS classification task, individually or in combination with other embeddings.

Performance of Text Embeddings. The upper section of Table 7.3 displays the results of various combinations of text embeddings. Incorporating both $\boldsymbol{cl}$ and $\boldsymbol{desc}$ consistently improves the performance for the InjVal dataset. While adding $\boldsymbol{cl}$ yields mixed outcomes for Rito and Atz, combining all three text embeddings generally produces favorable results, with the exception of the macro-F1 score for Atz. This aligns with the findings presented in Section 6.5, demonstrating that a document representation generated by combining all field embeddings using vector summation yields the best performance.

Performance of CPC-label Embeddings. The middle portion of Table 7.3 displays the results obtained using various CPC-label embeddings. Among these, SVM + $\boldsymbol{cpc}_{tf.idf}$ produces the highest macro- and micro-F1 scores for InjVal and Rito. Across datasets, embeddings from patent text exhibit superior performance compared to the best-performing model using only CPC information (SVM + $\boldsymbol{cpc}_{tf.idf}$), with one exception: for Rito, where the macro-F1 achieved using SVM + $\boldsymbol{cpc}_{tf.idf}$ outperforms that reached using TMM + $\boldsymbol{ta}$.

Among the more sophisticated CPC label feature vectors, TF-IDF with concatenated label descriptions ($\boldsymbol{cpc}_{tf.idf}$) performs the best, in terms of macro-F1, across all datasets. Furthermore, upon comparing the neural label embeddings across datasets, it can be observed that graph-based embeddings ($\boldsymbol{cpc}_{graph}$) consistently outperform description-based embeddings ($\boldsymbol{cpc}_{text}$).

Finally, to demonstrate that there is no one-to-one mapping between target labels and CPC labels in the PLS datasets, the performance of the multi-hot encoded vector $\boldsymbol{cpc}_{multihot}$ is evaluated as a sanity check. For InjVal, a high micro-F1 score with $\boldsymbol{cpc}_{multihot}$ indicates that CPC/IPC labels were used in the dataset creation process; however, the macro-F1 score is lower than it is for other CPC-label embeddings. While $\boldsymbol{cpc}_{multihot}$ is informative, its performance can be enhanced using more sophisticated CPC-label embeddings.

In Section 4.5, we analyzed the association between CPC/IPC labels and PLS-oriented categories by computing Point-wise Mutual Information (PMI) scores (Church and Hanks, 1990) for each CPC/IPC label and PLS category pair. We then analyzed and compared the PMI values for the three PLS datasets. This analysis was motivated by the hypothesis that a document representation method utilizing CPC-label embeddings would perform better for a dataset with a higher association between CPC/IPC labels and PLS-oriented categories. The analysis of PMI scores revealed that there is a stronger association between CPC/IPC labels and PLS-oriented categories in InjVal compared to the other two datasets. Furthermore, the findings presented in Table 7.3 indicate that the CPC-label embeddings perform better for the InjVal dataset than the other two PLS datasets, thereby confirming the hypothesis.

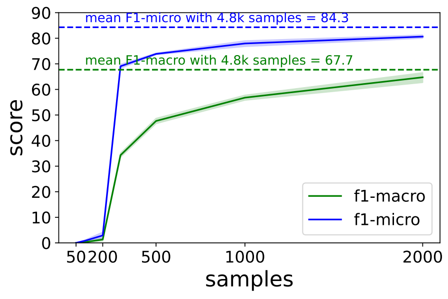
Model	InjVal macro-F1	InjVal micro-F1	Rito macro-F1	Rito micro-F1	Atz macro-F1	Atz micro-F1
Benites et al. (2018): SVM	61.4±2.1	74.1±4.3	51.1±6.8	58.2±2.5	65.4±1.8	71.9±2.5
Pujari et al. (2021a): TMM with $\boldsymbol{ta}$	65.2±2.1	79.2±1.3	44.3±4.0	66.0±3.0	62.1±2.2	70.6±1.1
TMM with $\boldsymbol{ta} \oplus \boldsymbol{cl}$	66.1±2.0	82.0±0.8	39.3±5.1	64.5±1.8	64.7±1.8	71.3±2.1
TMM with $\boldsymbol{ta} \oplus \boldsymbol{cl} \oplus \boldsymbol{desc}$	66.2±4.9	82.2±1.7	49.1±6.9	66.3±1.9	62.6±4.0	71.2±2.6
TMM with $\boldsymbol{cpc}_{multihot}$	49.8±3.7	77.8±0.9	17.3±2.5	42.0±8.0	23.8±5.3	39.9±5.2
TMM with $\boldsymbol{cpc}_{text}$	54.7±2.8	73.8±0.5	28.1±1.9	60.5±2.5	37.0±3.5	47.9±1.5
TMM with $\boldsymbol{cpc}_{graph}$	58.6±1.7	76.5±0.7	35.2±5.5	62.9±1.9	44.2±3.2	50.8±3.8
TMM with $\boldsymbol{cpc}_{tf.idf}$	60.4±1.9	75.9±0.8	39.2±6.0	60.5±3.9	44.4±2.5	52.8±2.2
SVM with $\boldsymbol{cpc}_{tf.idf}$	63.0±1.2	76.7±1.4	45.2±5.4	61.4±2.1	50.1±1.9	58.6±1.2
TMM with $\boldsymbol{ta} \oplus \boldsymbol{cl} \oplus \boldsymbol{desc}; \boldsymbol{cpc}_{tf.idf}$	66.6±0.5	83.9±0.4	46.4±5.7	65.1±2.6	63.4±2.8	71.1±1.1
TMM with $\boldsymbol{ta} \oplus \boldsymbol{cl} \oplus \boldsymbol{desc}; \boldsymbol{cpc}_{graph}$	**67.7**±2.5	**84.3**±0.5	**53.9**±6.6	**67.7**±3.2	**66.2**±1.8	**73.2**±2.1

Table 7.3: Comparison of text-based and CPC-based embeddings. Benites et al. (2018) use TF-IDF-based vectors for title, abstract, description, and claims.

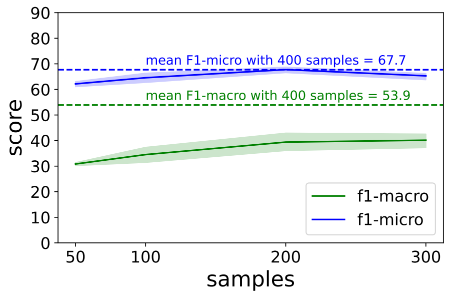
Dataset	Model	macro-avg. P	macro-avg. R	macro-avg. F1	micro-avg. P	micro-avg. R	micro-avg. F1
InjVal	Benites et al. (2018): SVM	64.0±3.8	**69.7**±7.1	61.4±2.1	61.7±7.0	**93.8**±2.3	74.1±4.3
InjVal	Pujari et al. (2021a): TMM with $\boldsymbol{ta}$	68.8±3.5	65.2±1.9	65.2±2.1	78.8±1.3	79.6±1.5	79.2±1.3
InjVal	TMM with $\boldsymbol{ta} \oplus \boldsymbol{cl} \oplus \boldsymbol{desc}; \boldsymbol{cpc}_{graph}$	**74.3**±6.7	66.4±1.6	**67.7**±2.5	**84.2**±0.8	84.4±0.6	**84.3**±0.5
Rito	Benites et al. (2018): SVM	46.6±13.0	**69.9**±4.6	51.1±6.8	43.1±3.3	**90.1**±2.3	58.2±2.5
Rito	Pujari et al. (2021a): TMM with $\boldsymbol{ta}$	58.5±5.0	42.2±4.6	44.3±4.0	67.8±2.9	64.3±3.9	66.0±3.0
Rito	TMM with $\boldsymbol{ta} \oplus \boldsymbol{cl} \oplus \boldsymbol{desc}; \boldsymbol{cpc}_{graph}$	**64.4**±6.9	49.5±6.5	**53.9**±6.6	**70.7**±2.6	65.1±4.9	**67.7**±3.2
Atz	Benites et al. (2018): SVM	66.7±7.8	**70.0**±5.9	65.4±1.8	65.2±6.4	**81.0**±4.4	71.9±2.5
Atz	Pujari et al. (2021a): TMM with $\boldsymbol{ta}$	68.6±1.4	59.7±4.0	62.1±2.2	73.0±2.6	68.8±4.4	70.6±1.1
Atz	TMM with $\boldsymbol{ta} \oplus \boldsymbol{cl} \oplus \boldsymbol{desc}; \boldsymbol{cpc}_{graph}$	**72.2±3.9**	63.3±1.5	**66.2**±1.8	**75.6**±4.1	70.9±1.9	**73.2**±2.1

Table 7.4: Comparison of our best-performing approach to the non-neural and neural baselines.

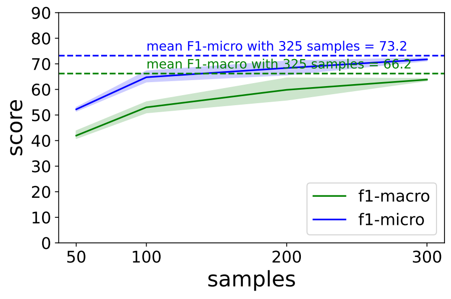
Performance When Combining CPC-label and Text Embeddings. When combining CPC embeddings with the patent text embeddings and using TMM, $\boldsymbol{cpc}_{graph}$ outperforms $\boldsymbol{cpc}_{tf.idf}$ (see bottom-most part of Table 7.3). While $\boldsymbol{cpc}_{graph}$ is directly trained as a dense embedding, combining $\boldsymbol{cpc}_{tf.idf}$ with the TMM model is not straightforward due to its high dimensionality. For scalability reasons, the $\boldsymbol{cpc}_{tf.idf}$ embedding is down-projected from 140k to 768 dimensions using a linear layer and then integrated into the TMM model. We hypothesize that this dimensionality reduction is responsible for the drop in performance. It can be concluded that a combination of all patent text field embeddings and $\boldsymbol{cpc}_{graph}$, i.e., TMM with $\boldsymbol{ta} \oplus \boldsymbol{cl} \oplus \boldsymbol{desc}; \boldsymbol{cpc}_{graph}$, is most effective for the PLS classification task across datasets.



(a) InjVal



(b) Ritonavir



(c) Atazanavir

Figure 7.3: The classification performance of the best-performing model, i.e., TMM with $\boldsymbol{ta} \oplus \boldsymbol{cl} \oplus \boldsymbol{desc}; \boldsymbol{cpc}_{graph}$, for varying numbers of training instances. As can be seen, the optimal classification performance is achieved with 200 to 300 instances.

7.5.3 Comparison with the Baselines

Table 7.4 shows that our best-performing approach (TMM with $\boldsymbol{ta} \oplus \boldsymbol{cl} \oplus \boldsymbol{desc}; \boldsymbol{cpc}_{graph}$) outperforms the baselines in terms of macro- and micro-F1 scores across the three datasets. The SVM model by Benites et al. (2018) excels in recall, but our method achieves much higher precision and, hence, higher macro- and micro-F1 scores. Note that the prediction threshold of the SVM model is optimized to maximize macro-F1 on the validation set.

By comparing TMM with $\boldsymbol{ta}$ to the neural baseline, we find that adding information from additional text fields and CPC embeddings improves performance. This finding is further supported by the fact that abstracts are often duplicated across patents (see Section 4.6); this is particularly likely to be the case when performing a PLS within a narrow field, for example, in relation to a specific invention. Therefore, it is of paramount importance to use additional textual content fields.

The proposed approach consistently outperforms the baselines across three datasets in terms of micro- and macro-F1 due to balanced precision and recall scores. We suggest that the proposed approach provides a robust method that may be used as a basis for future work and for real-world PLS classification tasks.

7.5.4 Minimum Training Instances

Motivated by the high cost of manual labeling by domain experts, we perform a study to determine the minimum number of training instances required to train a classification model that has acceptable performance with unseen data. Figure 7.3 shows macro-F1 and micro-F1 scores over different training sizes where the training instances were randomly sampled. Across datasets, an acceptable micro-F1 performance can be achieved with a training size of between 200 to 300 instances. On Rito and Atz, near-optimal micro-F1 is achieved with a training size of 200 instances. On the InjVal dataset, a training size of 300 leads to a micro-F1 of around 70, compared to the maximum micro-F1 score of 84.3 achieved with the complete dataset of 4.8k instances. These results illustrate that with as few as 200 instances systems can be developed that already have significant value for patent professionals. However, the lower macro-F1 score indicates insufficient performance for infrequent target labels; if this approach is applied in cases where target labels are infrequent, further research is needed to ensure high performance with minimal training data.

7.5.5 Summary of Results

We experimented with four different CPC label representations and found that the TF-IDF-based feature vector performs best by capturing crucial information using domain-specific keywords. Across three datasets, the best performance is achieved using a document representation that combines the best-performing patent text embedding with graph-based label representation. Further, we show that acceptable classification performance can be achieved with only 200-300 training instances. In the following, we evaluate the proposed classification technique on a similar task, that of classifying PubMed articles in the LitCovid dataset into categories relating to COVID-19 (Chen et al., 2021c).

7.6 Out-of-Domain Evaluation: Classifying PubMed Articles by COVID-19 Categories

The document representations proposed in Section 7.3 are evaluated on the PLS classification task in Section 7.5, where it is shown that a classifier's performance can be improved using a document representation that combines CPC labels and patent text. To demonstrate the applicability of the proposed approach beyond the context of patents, we evaluate it on a similar task: classifying PubMed articles to seven COVID-19-related categories. Specifically, we participated in Task 5 of the BioCreative VII challenge (Chen et al., 2021b), the objective of which was to automate the labeling of articles in the LitCovid document collection (Chen et al., 2021c). LitCovid is a corpus containing articles related to COVID-19, and automated labeling can facilitate the discovery of relevant articles.

Task 5 of the BioCreative challenge provides a dataset consisting of title and $\mathrm{abstract}$ as text fields for each publication, as well as metadata such as $\mathrm{pub\_type}$, $\mathrm{journal}$, and keywords. We experimented with various techniques using publication text and metadata to assign seven categories to COVID-19-related articles. The seven categories are “Treatment”, “Mechanism”, “Prevention”, “Case Report”, “Diagnosis”, “Transmission”, and “Epidemic Forecasting”, The task setting was similar to that described by Sadat and Caragea (2022), in which the labels corresponding to a classification scheme provided by the Association for Computing Machinery (ACM) were predicted, with a research publication as input. They model this as a multi-task learning problem, with two classification heads predicting ACM labels and author-provided keywords. We see two problems with their approach. Firstly, the author-provided keywords belong to a dynamic taxonomy since authors can add new keywords with a submission. Secondly, the same concept might be represented by more than one keyword. For example, “severe acute respiratory syndrome coronavirus 2” and “sars-2” refer to same indicator as “SARS-CoV-2”.[3]

Like CPC labels, keywords are an essential source of semantic information, and incorporating them into a document representation may improve classification performance. Preliminary experiments revealed that metadata encoded as multi-hot vectors are generally informative, and among them, the multi-hot representation of author-provided keywords performs best. Based on our assessment of the shortcomings of Sadat and Caragea (2022), we represent an instance by combining the SciBERT (Beltagy et al., 2019) embeddings for $\mathrm{title} + \mathrm{abstract}$ and concatenated text for keywords using vector concatenation (Figure 7.4). We observe that the best-performing classifier utilizes the Transformer-based Multi-task Model (TMM) proposed in Chapter 5 as the classification model. Compared to the ML-Net baseline (Du et al., 2019) provided by the task organizers, our proposed model performs better across all labels, improving macro-F1 by 13% in relative terms. Additionally, our system performs comparably to the third quartile (Q3) of both macro- and micro-F1 scores, indicating that it outperformed approximately 75% of competing submissions. This result demonstrates that even though our system was initially developed for patent classification, it is also effective in the biomedical domain on a task for classifying PubMed articles.

7.6.1 Related Work

The shared task dataset is taken from the LitCovid database (Chen et al., 2021c), a collection of PubMed articles relating to COVID-19 research. LitCovid is updated every day, with new documents being added with manually labeled categories. In addition, the task organizers provided ML-NET (Du et al., 2019) as a baseline, which uses a BiLSTM and ELMo for classifying biomedical documents. Using a transformer model pre-trained on the biomedical domain text, i.e., BioBERT (Lee et al., 2020), Gutierrez et al. (2020) demonstrated improved topic classification performance over the LitCovid dataset compared to BERT (Devlin et al., 2019). In a similar task, Sadat and Caragea (2022) propose a multi-task classification approach that predicts categories for scientific documents and

[3] https://en.wikipedia.org/wiki/Category:SARS-CoV-2 [last accessed December 10, 2023]

Metadata type	# of unique labels	Avg. per instance
pub_type	50	1.5
journal	4,251	1.0
keywords	35,766	4.2

Table 7.5: Statistics for the metadata fields in the LitCovid dataset.

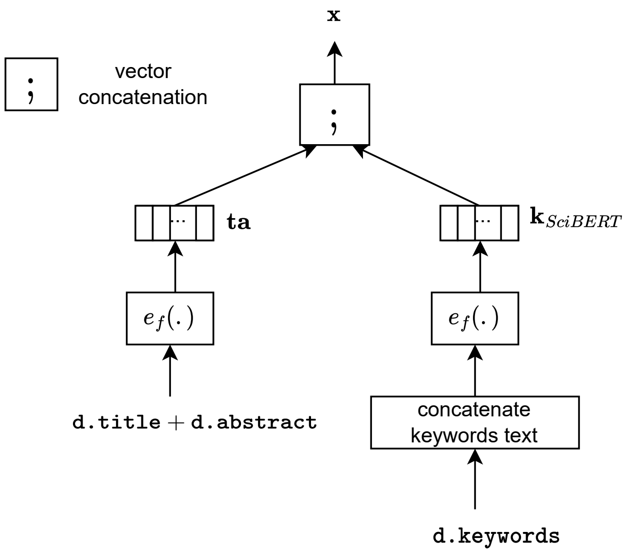


Figure 7.4: The document representation $\boldsymbol{x}$ is generated by combining the title and abstract embeddings $\boldsymbol{ta}$ with keyword text embedding $\boldsymbol{k}_{SciBERT}$ created with a SciBERT-based text representation method $e_f(.)$.

user-provided keywords. We take an approach different from Sadat and Caragea (2022) and incorporate the keywords directly into the document representation.

7.6.2 Methodology

The classification process follows a two-step approach. A publication is mapped to an n-dimensional feature vector; this is then provided as input to the classification model, which predicts a prediction score for each label. The LitCovid dataset provides title and abstract as content fields and pub_type, journal, and keywords as metadata. We explore different techniques for generating the feature vector by experimenting with metadata embeddings either alone or combined with the publication-text embedding.

To generate a publication-text embedding, we concatenate the title and abstract fields and input them to SciBERT (Beltagy et al., 2019). The sequence embedding generation process is denoted by the method $e_f(.)$ and is described in Section 2.4.3, and the resulting embedding is denoted as $\boldsymbol{ta}_{SciBERT}$.

For the metadata, pub_type, journal, and keywords are possible sources of information with which to generate a representation. We define the embeddings corresponding to the metadata as follows:

- As each article can only appear in a single journal, $\mathrm{journal}$ is encoded as a one-hot vector, $\boldsymbol{j}_{one-hot}$.

- Since multiple $\mathrm{pub\_type}$ values (e.g., "Journal Article", "Randomized Controlled Trial", or "Research Support Non-U.S. Gov't") might be associated with a document, $\mathrm{pub\_type}$ is represented as a multi-hot encoded vector, $\boldsymbol{p}_{multi-hot}$. The instances in the dataset are associated with 50 different publication types.

- A document might contain multiple author-provided keywords. Hence, keywords are represented as a multi-hot encoded vector, $\boldsymbol{k}_{multi-hot}$. Users are free to add any text as keywords, and this results in a relatively large vocabulary and, thus, a sparser feature vector as compared to $\boldsymbol{j}_{one-hot}$ and $\boldsymbol{p}_{multi-hot}$. Therefore, $\boldsymbol{k}_{multi-hot}$ is down-projected to 768 using a linear layer.

- The keyword vocabulary contains terms that refer to the same entity or are semantically similar, information which is not captured by $\boldsymbol{k}_{multi-hot}$. Moreover, the set of keywords is dynamic so that an author can add new keywords for their publication. For this reason, in addition to a multi-hot vector, we compute $\boldsymbol{k}_{SciBERT}$ by first concatenating keywords and then passing the concatenated text through a SciBERT model.

As with the PLS document representation methods, we experiment with individual embeddings and a representation generated by concatenating the embeddings using the semicolon operator (;). As discussed in Section 3.2, the classification pipeline takes an instance of the domain-specific document model, $\mathrm{PubMedDM}$ in this case, maps it to a domain-independent document model, which is finally mapped to an instance of the data model that is specific to a document representation method. Here, the best-performing document representation method ($\boldsymbol{ta}_{SciBERT}$; $\boldsymbol{k}_{SciBERT}$) takes an instance of $\mathrm{DocRepDM\_Seq\_Label}$ as input. $\mathrm{DocRepDM\_Seq\_Label}$ has text, $\mathrm{max\_length}$, and $\mathrm{source\_labels}$ as attributes, which hold the values for concatenated text from title and $\mathrm{abstract}$, maximum sequence length, and author-provided keywords, respectively (see Section 3.2.4). The document representation process is depicted in Figure 7.4. The architecture of our classification model is similar to the Transformer-based Multi-task Model (TMM) introduced in Chapter 5 and used earlier in this chapter for PLS classification (Section 7.3).

7.6.3 Experimental Setup

Below, we define the experimental setup and provide details on the evaluation metrics, hyperparameters, and baselines.

Dataset. The shared task dataset comprises 24,960 training instances and a validation set of 6,239 instances. During the evaluation process, the participating teams' submissions are assessed on a blind test set of 2,500 instances with unknown target labels. The evaluation of

	Label-based					
	macro-avg.			micro-avg.		
	P	**R**	**F1**	**P**	**R**	**F1**
TMM with $\boldsymbol{p}_{multi-hot}$	46.2	29.1	31.7	59.1	39.7	47.5
TMM with $\boldsymbol{j}_{multi-hot}$	59.6	33.4	41.5	70.6	45.9	55.6
TMM with $\boldsymbol{k}_{multi-hot}$	61.3	42.0	49.6	71.3	51.3	59.7
TMM with $\boldsymbol{k}_{SciBERT}$	**69.7**	**49.4**	**57.7**	**78.3**	**57.8**	**66.5**

Table 7.6: Results for metadata-based document representations.

the submissions on the test set generates task statistics. To fine-tune the model parameters and select the best-performing model with early stopping, we split the training data into a training set containing 22,464 instances and a validation set with a size of 2,496. The validation set is employed to assess the submitted runs.

Evaluation Metrics. As classifying PubMed articles is a flat classification task similar to PLS classification, we evaluate the classifier's output using precision, recall, and F1 as evaluation metrics. For each evaluation metric, we compute both micro- and macro-average scores (see Section 2.7). The micro-average score is computed over all the instances in the test set and provides an overall estimate of a model's performance. The macro-average score is an average of the performance over all the labels, thus indicating a model's performance for infrequent class labels as well.

Hyperparameters. The vector dimensions for $\boldsymbol{p}_{multi-hot}$, $\boldsymbol{j}_{one-hot}$, and $\boldsymbol{k}_{multi-hot}$ are equal to the respective number of possible values, resulting in sizes of 50, 4251, and 35,766, respectively (Table 7.5). The best-performing document representation concatenates $\boldsymbol{ta}_{SciBERT}$ and $\boldsymbol{k}_{SciBERT}$, resulting in a 1536-dimensional vector. We use a learning rate of 1e-5, dropout rate of 0.25, and a batch size of 64. Each of the models is trained on a single Nvidia Tesla V100 GPU. Also, we apply early stopping if the macro-F1 score on the validation set does not improve for five epochs. The transformer-based models take 100 to 120 hours to train, whereas the models using only metadata information can be trained in one to two hours.

Baseline. The task organizers provide ML-Net (Du et al., 2019) as a baseline. To represent the document text, ML-Net uses an ELMo (Peters et al., 2018) to generate a sequence of token embeddings, which are then aggregated using a BiLSTM (Schuster and Paliwal, 1997) and an attention layer to form a document vector. Finally, the classification model predicts the label score and label count for a document vector and then uses these values to predict the target label set.

		macro-avg.			micro-avg.		
		P	**R**	**F1**	**P**	**R**	**F1**
Task Statistics	Mean	86.7	80.1	81.9	89.7	86.2	87.8
	Std	6.0	7.9	7.0	5.4	4.8	4.3
	Q1	84.6	75.5	76.5	88.0	84.5	85.4
	Median	88.4	83.9	85.3	91.1	88.4	89.3
	Q3	**90.8**	85.6	**86.7**	**92.5**	89.7	**90.8**
Models	Baseline (ML-Net) Du et al. (2019)	83.6	73.1	76.6	87.6	81.4	84.4
	TMM with $\boldsymbol{ta}_{SciBERT}$	85.7	86.1	85.6	91.2	88.9	90.0
	TMM with $\boldsymbol{ta}_{SciBERT}; \boldsymbol{k}_{multi-hot}$	84.7	**87.3**	85.4	89.3	**90.2**	89.7
	TMM with $\boldsymbol{ta}_{SciBERT}; \boldsymbol{k}_{SciBERT}$	89.0	84.8	86.6	92.2	88.6	90.3

Table 7.7: Comparing different versions of the best-performing model to baseline and task statistics.

7.6.4 Results

First, we evaluate the performance of different metadata embeddings in classifying the publications into COVID-19-related categories, and then we combine the best-performing metadata embedding with the publication-text embedding, i.e., $\boldsymbol{ta}_{SciBERT}$.

Comparing Metadata and Publication-Text Embeddings. Table 7.6 reports our results on the validation set when using one document representation vector at a time. We find that among the metadata, $\mathrm{keywords}$ are most informative with regard to the classification task and that the SciBERT-based embedding outperforms the multi-hot encodings.

Comparison with the Baselines. Table 7.7 reports results on the blind test set using the shared task statistics provided by the task organizers. The last three rows show the results of our submissions with three different document representation settings, which are defined as follows:

- $\boldsymbol{ta}_{SciBERT}$ is generated by embedding the title and $\mathrm{abstract}$ with SciBERT.
- $\boldsymbol{ta}_{SciBERT}$;$\boldsymbol{k}_{multi-hot}$ is generated by concatenating the SciBERT-encoded title and $\mathrm{abstract}$ with the multi-hot encoded keyword vector.
- $\boldsymbol{ta}_{SciBERT}$;$\boldsymbol{k}_{SciBERT}$ is generated by concatenating the SciBERT-encoded title and $\mathrm{abstract}$ with the SciBERT-encoded keywords embedding.

Among the different representation techniques, $\boldsymbol{ta}_{SciBERT}$;$\boldsymbol{k}_{SciBERT}$ performs best on the blind test set, achieving a score that is on par with the Q3 statistics for the task, meaning that it is better than roughly 75% of the systems submitted for the challenge. In addition, we observe that all our submitted runs perform better than the ML-Net baseline (Du et al., 2019).

Summary. We participated in Task 5 of the BioCreative VII challenge and proposed a model architecture consisting of TMM as a classification model and a document representation combining the publication's text and keywords, which performed much better than the baseline system. This evaluation demonstrates the applicability of the approach we propose for PLS in a classification setting other than patents.

7.7 Conclusions

To promote research on automating patent landscape studies, we compare various neural and non-neural methods using input representations encompassing patent texts and CPC information. As a result, we propose a competitive neural patent classification model that harnesses both patent text and CPC label information and demonstrates robust performance across all three datasets. We discovered that acceptable performance in terms of micro-F1 can be achieved with as few as 200 to 300 training instances, highlighting the practical applicability of the proposed approach. To enhance performance for infrequent classes, future research could explore the integration of our methods with active learning and few-shot techniques. We demonstrate the applicability of our approach in a similar task setting other than patents by evaluating it on the task of assigning COVID-19 categories to PubMed articles.

Chapter 8

Summary and Outlook

This thesis introduces novel patent classifiers that address the challenges associated with patents. The proposed techniques aim to reduce the manual effort required to perform patent categorization tasks in various business-critical use cases. Here, we summarize the key challenges, highlight our contributions, and discuss the limitations of our work.

8.1 Summary and Conclusion

In this thesis, we focus on two patent classification tasks: CPC classification and PLS classification (see Definitions 2.1 and 2.2, page 18), with potential for the application of this work to other patent use cases, such as prior art search and patent alert systems. We curate and release datasets for both tasks, which we utilize to conduct crucial analyses that inform the design of novel document representation methods and classification models. In this regard, we summarize our main contributions as follows:

(1) Conceptual Model for Classification. One of the requirement engineering challenges is to adapt the classification pipeline for new use cases and novel classification techniques by expending minimal effort. A conceptual model can be valuable when designing and adapting a business solution that has a machine learning system as one of the components (Maass and Storey, 2021). A conceptual model offers stakeholders a unified system perspective, illustrating different components and their interactions. As one of the main contributions of this thesis, in Chapter 3, we propose a conceptual model for document classification based on the GR4ML framework (Nalchigar et al., 2021). Further, we expand this conceptual model by defining a classification pipeline that can be adapted to various classification tasks and novel document representation methods.

(2) Patent Classification Datasets. To the best of our knowledge, no dataset currently exists for the PLS classification task. As one of our significant contributions, we curate three datasets for PLS from two diverse domains (see Section 4.4). Two of these datasets are based on PLSs conducted by the World Intellectual Property Organization (WIPO), which we have further curated and enriched with additional fields obtained from PatBase.[1] The third dataset, Injection Valve, consists of patents that were labeled by a domain expert over the course of 25 years. Furthermore, the existing datasets (Li et al., 2018a; Lee and Hsiang, 2019) for CPC classification only utilize a limited number of patent fields, namely the patent's title, abstract, and claims, excluding the more detailed description section. As another of our major contributions to open-source datasets, we release the USPTO-70k dataset to enable CPC classification research using the full texts of patents (see Section 4.3).

(3) Analysis on Duplicate Patent Text. Patents possess a unique characteristic whereby text chunks, such as sentences and paragraphs, can be duplicated within and across documents. To the best of our knowledge, no concrete studies have analyzed the duplication of text in patents. As a significant contribution of Chapter 4, we analyze duplicate sentences and abstracts. We find that 15% of the sentences in the USPTO-70k dataset are duplicated within and across patents, whereas 17% of the abstracts in the USPTO-7M dataset are duplicated (see Section 4.6). This analysis of duplicate text emphasizes the redundant nature of patents and motivates us to explore patent representation techniques that select a subset of the most informative semantic elements for representation.

(4) Transformer-Based Hierarchical Multitask Model. Although CPC classification is a hierarchical task, previous works have predominantly treated it as a flat classification task, often focusing on predicting labels at the third level of the hierarchy (Li et al., 2018a; Lee and Hsiang, 2019; Zaheer et al., 2020). However, a hierarchical classifier that learns to predict labels at different granularity levels may perform better in hierarchical classification tasks. In this context, we draw inspiration from a non-neural hierarchical classification technique based on the local classifier per node approach (Benites, 2019). This technique involves training a classifier for each label in the taxonomy and utilizing the hierarchical taxonomic information for prediction. However, training such a model with a classifier for each label using a transformer-based language model is infeasible due to memory constraints. To address this issue, as one of our significant contributions, we propose a memory-efficient local classifier per node approach called the Transformer-based Multitask Model (TMM). TMM utilizes a shared transformer-based language model for text representation and trains a classification head for each label (see Section 5.3). The proposed technique outperforms a variety of neural and non-neural baselines (Li et al., 2018a; Lee and Hsiang, 2019; Huang et al., 2019; Benites, 2019). Furthermore, we introduce a hierarchical variant of TMM, the Transformer-based Hierarchical Multitask Model (THMM). THMM leverages the hierarchical links in the taxonomy to transfer the learned representation from a parent head to the heads corresponding to its children. Our analysis demonstrates that THMM performs better for less frequent labels at the lower levels of the CPC taxonomy due to these hierarchical links.

[1]https://www.patbase.com/express/login.asp [last accessed December 10, 2023]

(5) Efficient Patent Representation Using Full Patent Text. Patents are known for their extensive length, which poses a challenge for text representation techniques, particularly for transformer-based models that restrict input length to 512 tokens, such as BERT (Devlin et al., 2019). Moreover, our analysis of duplicate text underscores the redundant nature of the patent text. Therefore, in Chapter 6, we propose document representation techniques that address these challenges by selecting a subset of the most informative text elements and then utilizing them to generate a document representation using a transformer model (see Section 6.3). As a significant contribution, we propose a document representation technique that combines patent fields in order of their informativeness, as determined by the performance of the corresponding text field on the CPC classification task. The training process is efficient as the SciBERT model fine-tuned over the CPC classification task in Chapter 5 is further used to generate text embeddings without any further fine-tuning of SciBERT parameters. Additionally, we introduce a novel sentence ranker designed to select the most informative sentences from a patent. Our evaluation demonstrates that optimal classification accuracy can be achieved by utilizing a subset of the most informative text elements (e.g., fields, sentences, and tokens).

(6) Neural Representations for Patent Landscape Study. Patent Landscape Studies present a unique document classification scenario in which a training instance is labeled with CPC/IPC labels and PLS-oriented categories. The labels in the two label sets are taken from two different taxonomies. When testing the model, the CPC/IPC labels are known and can be utilized to enrich document representation and predict PLS-oriented categories as target labels. Our experiments demonstrate that incorporating CPC label information improves prediction accuracy, and the proposed technique exhibits robust performance across diverse PLS datasets (see Section 7.5). Furthermore, we demonstrate the applicability of the proposed technique in domains other than patents through our evaluation of an out-of-domain classification of PubMed articles into seven categories relating to COVID-19 (see Section 7.6).

8.2 Outlook and Discussion

Above, we have highlighted the most important contributions of this thesis to automating the patent classification problem and addressing key challenges. However, it is essential to acknowledge that the problem is far from being completely solved. In this section, we will discuss the limitations of our work, other existing challenges that have not been addressed, and potential future directions to explore.

(1) Applying LLMs for Patent Classification. Brown et al. (2020) propose the Generative Pre-trained Transformer model version 3 (GPT-3) with 175 billion parameters, which is trained with 570 GB data filtered from the Common Crawl dataset.[2] They demonstrate

[2]https://commoncrawl.org/ [last accessed December 10, 2023]

that when scaled to hundreds of billions of parameters and trained with a large amount of text data, the large language models (LLMs) acquire the ability to perform considerably better in unseen task settings in zero-shot and few-shot scenarios, even outperforming fine-tuned models in some cases. To further enhance the ability of GPT-3 to follow instructions, Ouyang et al. (2022) train a version of the GPT-3 model on human-generated instruction using Reinforcement Learning from Human Feedback (RLHF) technique (Christiano et al., 2017). They showed that the power of LLMs could be harnessed without further fine-tuning the model. This is achieved with in-context learning, a method that invokes the task-specific capabilities of a model when a task description, a few examples, and an input text are provided as prompts.

However, the applicability of in-context learning in a multi-label setting with long text documents is limited for the following reasons. First, the few-shot in-context learning is infeasible due to the limited context length of the input prompt for LLMs. Patents contain 12.5k tokens on average (see Figure 4.5, page 61), and the CPC/IPC taxonomy consists of more than 600 subclass labels (see Section 2.1.3). Second, evaluating each document text and label description for entailment, similar to Shen et al. (2021), might incur considerable costs when hundreds of labels are involved, e.g., the CPC/IPC classification task. Nevertheless, we find a few possible research directions for using LLMs for multi-label classification, as described below.

Pu et al. (2023) show that summaries generated using LLMs are comparable to or better than the human annotators. Further, with our evaluation in Section 6.5.2, we find that the extractive summarization methods effectively identify the most informative sentences from patents for the classification task. Given the redundant nature of texts in patents and the ability of LLMs to effectively identify informative document text, we see the possibility of employing LLMs for efficient classification of long text by performing summarization before inference. As another related research direction, the LLMs can be employed for data augmentation, particularly for class imbalance scenarios (Wang et al., 2023).

However, model fine-tuning is possible in relatively smaller models. Following the success of GPT-3 and its variants, Meta AI[3] released an open-source Large Language Model Meta AI (LLaMA) model in three different variants, which primarily differ in the number of parameters as 7B, 13B, and 70B, respectively (Touvron et al., 2023). Unlike the LLaMA-70B and GPT-3 models, it is feasible to fine-tune the LLaMA-7B and LLaMA-13B models on a single Nvidia A100 GPU using Parameter-Efficient Fine-Tuning (PEFT) methods, such as Low-Rank Adaptation (Hu et al., 2021), and Quantized Low-Rank Adaptation (Dettmers et al., 2023).

(2) Label-Based Document Representation. CPC/IPC labels play a crucial role in representing important domain concepts, and their descriptions provide valuable information associated with each label. In Chapter 6, label embeddings derived from the CPC/IPC label descriptions are used to select the most important sentences from a patent, whereas, in

[3]https://ai.meta.com/ [last accessed December 10, 2023]

Chapter 7, CPC/IPC label information is directly incorporated into the document representation, alongside the patent text, to predict PLS-oriented categories. In the first case, we use the target label description to select informative sentences from a document. In contrast, in the second case, the CPC/IPC embeddings are generated using sequence- and graph-based representation methods and incorporated into the document representation to predict PLS-oriented categories. Thus, incorporating CPC/IPC label information into document representation is an important research direction for CPC/IPC and PLS classification tasks (Niu and Cai, 2019).

(3) Few-Shot and Zero-Shot Classification. Following previous works targeting the CPC classification task, we evaluate the ability of the classifier to predict labels up to the third level of the hierarchy. This task reflects the process of pre-classification by which patent applications are assigned to the correct department within the examination office. However, the CPC/IPC class hierarchy extends to the fourth level and below, and these lower levels consist of a relatively high number of labels compared to the first three levels. The fourth level alone contains more than 10k labels, whereas the subgroup category, which includes labels from the fifth level and below, contains over 250k labels. These two label sets provide an ideal test bed for few-shot and zero-shot classification settings in which the training set contains very few or no training examples for a label.

The THMM model architecture described in Chapter 5 is defined for labels up to the third level of the CPC taxonomy. To predict labels at the fourth level, we can extend the THMM architecture using a local classifier per level approach for the labels in the fourth level. On the other hand, since most labels at the fifth level have very few or no labeled examples, exploring generative models for extreme multi-label classification might be a beneficial direction for further research (Zhang et al., 2023).

(4) Non-standard Terminology in Patents. Patent attorneys often employ a significant amount of non-standard terminology when writing patent text, particularly text for the claims field, with the intention of expanding the scope of inventions and obscuring information. For example, in the patent US20050089604A1[4], an invention related to an ice cream chip is described as "*An edible crisp unitary pastry having a double-curvature and having a planar longitudinal axis and a planar latitude axis perpendicular to the longitudinal axis*". Such use of non-standard terminology poses challenges when linking semantically similar multi-word terms and comparing inventions that employ diverse terminology. In this thesis, we employ SciBERT for text representation. SciBERT is a BERT variant pre-trained on the scientific text and exhibits a higher vocabulary overlap with patent terminology than BERT. Another important next step would be to link semantically similar multi-word terms. We consider this an important area for future research, as addressing this issue could improve the performance of patent retrieval and classification methods (Lyu et al., 2018; Nordquist and Meyers, 2022).

[4]https://patents.google.com/patent/US20050089604A1 [last accessed December 10, 2023]

(5) Long-Text Representation. Transformer-based language models generally perform better than the neural and non-neural text representation techniques that existed before the introduction of the transformer technique by Vaswani et al. (2017). The key transformer-based text representation methods can be grouped into three broad categories based on sequence length. The first group comprises methods that use the full-attention mechanism with a restricted input length of 512 tokens (Devlin et al., 2019; Liu et al., 2019). By employing sparse attention, the techniques comprising the second group can represent sequences of up to 4k tokens (Beltagy et al., 2020; Zaheer et al., 2020). The third group comprises techniques capable of representing sequences of arbitrary length during inference (Bertsch et al., 2023). Given the wide range of available options, evaluating their computational efficiency and classification accuracy is crucial to guide their applicability for a patent classification task.

(6) Temporal Characteristics of Patent Text and Taxonomy. As domain terminology evolves and the association between terms and labels changes over time, it is necessary to update classifiers. The USPTO-70k dataset, introduced in Chapter 5, aims to mimic the real-world setting with temporal data splits (see Section 4.3). However, it is essential to note that the experimental setting differs from the intended usage in the examination office, where patent applications and grants are added to the document collection on a weekly basis. When targeting real-world usage, it is crucial to evaluate the impact of concept drift on a classifier's performance (D'hondt et al., 2014).

As the domain evolves, the set of associated concepts changes, and therefore, labels are often added to or dropped from the label set; this necessitates adaptation of the classification model architecture.[5] Therefore, in addition to evaluation metrics, another important evaluation criterion for CPC classification is the robustness of classification model architectures to changes in the target label set. Compared to flat classification approaches (Li et al., 2018a; Lee and Hsiang, 2019), the TMM model can better adapt to a change in taxonomy. TMM consists of multiple classification heads, one for each label in the taxonomy. This means that if the taxonomy changes, the heads corresponding to the affected labels can be added or removed from the model.

Chapter Summary. With this thesis, we significantly contribute towards addressing the critical research challenges associated with patent classification tasks performed in business-critical applications. We curate and release new patent datasets and analyze data to identify critical challenges associated with patent classification. In addition, we propose novel classification methods that generate a document representation using both the patent text and label information and leverage the hierarchical taxonomy structure. We are optimistic that the proposed techniques can be successfully applied to other domains and document types by adapting the proposed classification pipeline with a few minor modifications.

[5]https://www.uspto.gov/web/patents/classification/cpc/html/cpc-notices-of-changes.html [last accessed December 10, 2023]

List of Figures

List of Tables

Bibliography

Louay Abdelgawad, Peter Kluegl, Erdan Genc, Stefan Falkner, and Frank Hutter. Optimizing Neural Networks for Patent Classification. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, Part III (ECML-PKDD '19)*, pages 688–703. Springer, 2019.

Aaron Abood and Dave Feltenberger. Automated Patent Landscaping. *Artificial Intelligence Law*, 26(2):103–125, 2018.

Doreen Alberts, Cynthia Barcelon Yang, Denise Fobare-DePonio, Ken Koubek, Suzanne Robins, Matthew Rodgers, Edlyn Simmons, and Dominic DeMarco. Introduction to Patent Searching. In *Current Challenges in Patent Information Retrieval*, volume 29 of *The Information Retrieval Series*, pages 3–43. Springer, 2011.

Juan Alcácer, Michelle Gittelman, and Bhaven Sampat. Applicant and Examiner Citations in U.S. Patents: An Overview and Analysis. *Research Policy*, 38(2):415–427, 2009.

Sophia Althammer, Mark Buckley, Sebastian Hofstätter, and Allan Hanbury. Linguistically Informed Masking for Representation Learning in the Patent Domain. In *Proceedings of the 2nd Workshop on Patent Text Mining and Semantic Technologies (PatentSemTech '21) co-located with the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21)*, 2021a.

Sophia Althammer, Sebastian Hofstätter, and Allan Hanbury. Cross-Domain Retrieval in the Legal and Patent Domains: A Reproducibility Study. In *Proceedings of the 43rd European Conference on Advances in Information Retrieval (ECIR '21)*, pages 3–17. Springer, 2021b.

Linda Andersson, Mihai Lupu, João R. M. Palotti, Allan Hanbury, and Andreas Rauber. When is the Time Ripe for Natural Language Processing for Patent Passage Retrieval? In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management (CIKM '16)*, pages 1453–1462. ACM, 2016.

Emmanuel Anguiano-Hernández, Luis Villaseñor Pineda, Manuel Montes-y-Gómez, and Paolo Rosso. Summarization as Feature Selection for Document Categorization on Small Datasets. In *Proceedings of the 7th International Conference on Advances in Natural Language Processing (IceTAL '10)*, pages 39–44. Springer, 2010.

Jean-Philippe Aumasson and Daniel J. Bernstein. SipHash: A Fast Short-Input PRF. Cryptology ePrint Archive, Paper 2012/351, 2012. URL https://eprint.iacr.org/2012/351. [last accessed August 16, 2023].

Alan David Baddeley. Working Memory. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 302(1110):311–324, 1983.

Siddhartha Banerjee, Cem Akkaya, Francisco Perez-Sorrosal, and Kostas Tsioutsiouliklis. Hierarchical Transfer Learning for Multi-label Text Classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL '19)*, pages 6295–6300. ACL, 2019.

Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. Scaling Up All Pairs Similarity Search. In *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*, pages 131–140. ACM, 2007.

Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP '19)*, pages 3613–3618. ACL, 2019.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. *CoRR*, abs/2004.05150, 2020.

Fernando Benites. TwistBytes - Hierarchical Classification at GermEval 2019: Walking the Fine Line (of Recall and Precision). In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS '19)*. German Society for Computational Linguistics & Language Technology, 2019.

Fernando Benites, Shervin Malmasi, and Marcos Zampieri. Classifying Patent Applications with Ensemble Methods. In *Proceedings of the Australasian Language Technology Association Workshop (ALTA '18)*, pages 89–92, 2018.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. Unlimiformer: Long-Range Transformers with Unlimited Length Input. *CoRR*, abs/2305.01625, 2023.

James E Bessen and Michael J Meurer. The Private Costs of Patent Litigation. In *Proceedings of the 2nd Annual Conference on Empirical Legal Studies Paper*. Boston University School of Law, 2008.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

Andrei Z. Broder. On the Resemblance and Containment of Documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, pages 21–29. IEEE, 1997.

Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-Wise Independent Permutations (Extended Abstract). In *Proceedings of the 13th Annual ACM Symposium on the Theory of Computing (STOC '98)*, pages 327–336. ACM, 1998.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models Are Few-Shot Learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20)*. Curran Associates Inc., 2020.

Linkun Cai, Yu Song, Tao Liu, and Kunli Zhang. A Hybrid BERT Model That Incorporates Label Semantics via Adjustive Attention for Multi-Label Text Classification. *IEEE Access*, 8:152183–152192, 2020.

Mosahid Khan Carsten Fink and Hao Zhou. Exploring the Worldwide Patent Surge. *Economics of Innovation and New Technology*, 25(2):114–142, 2016.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. Large-Scale Multi-Label Text Classification on EU Legislation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, Volume 1: Long Papers (ACL '19)*, pages 6314–6322. ACL, 2019.

Ilias Chalkidis, Xiang Dai, Manos Fergadiotis, Prodromos Malakasiotis, and Desmond Elliott. An Exploration of Hierarchical Attention Transformers for Efficient Long Document Classification. *CoRR*, abs/2210.05529, 2022.

Soumya Chatterjee, Ayush Maheshwari, Ganesh Ramakrishnan, and Saketha Nath Jagarlapudi. Joint Learning of Hyperbolic Label Embeddings for Hierarchical Multi-label Classification. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume (ACL '21)*, pages 2829–2841. ACL, 2021.

Fenxiao Chen, Yun-Cheng Wang, Bin Wang, and C.-C. Jay Kuo. Graph Representation Learning: A Survey. *APSIPA Transactions on Signal and Information Processing*, 9: e15, 2020.

Haibin Chen, Qianli Ma, Zhenxi Lin, and Jiangyue Yan. Hierarchy-aware Label Semantics Matching Network for Hierarchical Text Classification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Volume 1: Long Papers (ACL-IJCNLP '21)*, pages 4370–4379. ACL, 2021a.

Lei Chen, Houwei Chou, and Xiaodan Zhu. Developing Prefix-Tuning Models for Hierarchical Text Classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track (EMNLP '22)*, pages 390–397. ACL, 2022.

Qingyu Chen, Alexis Allot, Robert Leaman, Rezarta Islamaj Dogan, and Zhiyong Lu. Overview of the BioCreative VII LitCovid Track: Multi-label Topic Classification for COVID-19 Literature Annotation. In *Proceedings of the 7th BioCreative Challenge Evaluation Workshop*. (biocreative.bioinformatics.udel.edu), 2021b.

Qingyu Chen, Alexis Allot, and Zhiyong Lu. LitCovid: An Open Database of COVID-19 Literature. *Nucleic Acids Research*, 49(D1):D1534–D1540, 2021c.

Yen-Liang Chen and Yuan-Che Chang. A Three-Phase Method for Patent Classification. *Information Processing and Management*, 48(6):1017–1030, 2012.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers. *CoRR*, abs/1904.10509, 2019.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of the 8th Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8 '14)*, pages 103–111. ACL, 2014.

Seokkyu Choi, Hyeonju Lee, Eunjeong Park, and Sungchul Choi. Deep Learning for Patent Landscaping using Transformer and Graph Embedding. *Technological Forecasting and Social Change*, 175:121413, 2022.

Francois Chollet et al. Keras, 2015. URL https://github.com/fchollet/keras. [last accessed December 10, 2023].

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep Reinforcement Learning from Human Preferences. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS '17)*, page 4302–4310. Curran Associates Inc., 2017.

Kenneth Ward Church and Patrick Hanks. Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

Nigel S. Clarke, Björn Jürgens, and Victor Herrero-Solana. Blockchain Patent Landscaping: An Expert Based Methodology and Search Query. *World Patent Information*, 61: 101964, 2020.

Iain M. Cockburn and Megan J. Macgarvie. Patents, Thickets and the Financing of EarlyStage Firms: Evidence from the Software Industry. *Journal of Economics & Management Strategy*, pages 729–773, 2009.

Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20 (3):273–297, 1995.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. Overview of the TREC 2019 Deep Learning Track. *CoRR*, abs/2003.07820, 2020.

Xiang Dai, Ilias Chalkidis, Sune Darkner, and Desmond Elliott. Revisiting Transformer-Based Models for Long Document Classification. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7212–7230. ACL, 2022.

Jesse Davis and Mark Goadrich. The Relationship between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*, page 233–240. ACM, 2006.

Cinthia Mikaela de Souza, Magali R. G. Meireles, and Paulo Eduardo Maciel de Almeida. A comparative study of abstractive and extractive summarization techniques to label subgroups on patent dataset. *Scientometrics*, 126(1):135–156, 2021.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *Journal of American Society on Information Sciences*, 41(6):391–407, 1990.

Bart Degroote and Pierre Held. Analysis of the Patent Documentation Coverage of the CPC in Comparison with the IPC with a Focus on Asian Documentation. *World Patent Information*, 54:S78–S84, 2018.

Zhongfen Deng, Hao Peng, Dongxiao He, Jianxin Li, and Philip Yu. HTCInfoMax: A Global Model for Hierarchical Text Classification via Information Maximization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '21)*, pages 3259–3265. ACL, 2021.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. *CoRR*, abs/2305.14314, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies (NAACL-HLT '19), Volume 1 (Long and Short Papers), pages 4171–4186. ACL, 2019.

Eva D'hondt, Suzan Verberne, Cornelis H. A. Koster, and Lou Boves. Text Representations for Patent Classification. *Computational Linguistics*, 39(3):755–775, 2013.

Eva D'hondt, Suzan Verberne, Nelleke Oostdijk, Jean Beney, Cornelis H. A. Koster, and Lou Boves. Dealing with Temporal Variation in Patent Categorization. *Information Retrieval*, 17(5-6):520–544, 2014.

Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. CogLTX: Applying BERT to Long Texts. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20)*. Curran Associates Inc., 2020.

Rezarta Islamaj Dogan, Sun Kim, Andrew Chatr-aryamontri, Chih-Hsuan Wei, Donald C. Comeau, Rui Antunes, Sérgio Matos, Qingyu Chen, Aparna Elangovan, Nagesh C. Panyam, Karin Verspoor, Hongfang Liu, Yanshan Wang, Zhuang Liu, Berna Altinel, Zehra Melce Hüsünbeyi, Arzucan Özgür, Aris Fergadis, Chen-Kai Wang, Hong-Jie Dai, Tung Tran, Ramakanth Kavuluru, Ling Luo, Albert Steppi, Jinfeng Zhang, Jinchan Qu, and Zhiyong Lu. Overview of the BioCreative VI Precision Medicine Track: Mining Protein Interactions and Mutations for Precision Medicine. *Database Journal of Biological Databases Curation*, 2019:bay147, 2019.

Yanru Dong, Peiyu Liu, Zhenfang Zhu, Qicai Wang, and Qiuyue Zhang. A Fusion Model-Based Label Embedding and Self-Interaction Attention for Text Classification. *IEEE Access*, 8:30548–30559, 2020.

Zican Dong, Tianyi Tang, Junyi Li, and Wayne Xin Zhao. A Survey on Long Text Modeling with Transformers. *CoRR*, abs/2302.14502, 2023.

Jingcheng Du, Qingyu Chen, Yifan Peng, Yang Xiang, Cui Tao, and Zhiyong Lu. ML-Net: Multi-label Classification of Biomedical Texts with Deep Neural Networks. *Journal of the American Medical Informatics Association*, 26(11):1279–1285, 2019.

Christophe Dumet, Jérémy Pottier, Valérie Gouilleux-Gruart, and Hervé Watier. Insights Into the IgG Heavy Chain Engineering Patent Landscape as Applied to IgG4 Antibody Development. *mAbs*, 11(8):1341–1350, 2019.

Wafaa S. El-Kassas, Cherif R. Salama, Ahmed A. Rafea, and Hoda K. Mohamed. Automatic Text Summarization: A Comprehensive Survey. *Expert Systems with Applications*, 165:113679, 2021.

Jeffrey L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, 1990.

Caspar J. Fall, A. Törcsvári, K. Benzineb, and G. Karetka. Automated Categorization in the International Patent Classification. *SIGIR Forum*, 37(1):10–25, 2003.

Lintao Fang, Le Zhang, Han Wu, Tong Xu, Ding Zhou, and Enhong Chen. Patent2Vec: Multi-view Representation Learning on Patent-Graphs for Patent Classification. *World Wide Web*, 24(5):1791–1812, 2021.

Weiqi Feng and Dong Deng. Allign: Aligning All-Pair Near-Duplicate Passages in Long Texts. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21)*, pages 541–553. ACM, 2021.

Mario Franzosi. Novelty and Non-Obviousness: The Relevant Prior Art. *Journal of World Intellectual Property*, 3:683, 2000.

Annemarie Friedrich, Heike Adel, Federico Tomazic, Johannes Hingerl, Renou Benteau, Anika Marusczyk, and Lukas Lange. The SOFC-Exp Corpus and Neural Approaches to Information Extraction in the Materials Science Domain. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL '20)*, pages 1255–1268. ACL, 2020.

Lukas Galke and Ansgar Scherp. Bag-of-Words vs. Graph vs. Sequence in Text Classification: Questioning the Necessity of Text-Graphs and the Surprising Strength of a Wide MLP. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers (ACL '22)*, pages 4038–4051. ACL, 2022.

Alexander V Giczy, Nicholas A Pairolero, and Andrew A Toole. Identifying Artificial Intelligence (AI) Invention: A Novel AI Patent Dataset. *The Journal of Technology Transfer*, 47(2):476–505, 2022.

Mark Giereth. *An Architecture for Visual Patent Analysis*. PhD thesis, University of Stuttgart, 2013. URL http://elib.uni-stuttgart.de/opus/volltexte/2013/8021/. [last accessed December 10, 2023].

Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity Search in High Dimensions via Hashing. In *Proceedings of 25th International Conference on Very Large Data Bases (VLDB '99)*, pages 518–529. Morgan Kaufmann Publishers Inc., 1999.

Juan-Carlos Gomez. Analysis of the Effect of Data Properties in Automated Patent Classification. *Scientometrics*, 121(3):1239–1268, 2019.

Juan-Carlos Gomez and Marie-Francine Moens. A Survey of Automated Hierarchical Classification of Patents. In *Professional Search in the Modern World*, pages 215–249. Springer, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

Mattyws F. Grawe, Claudia A. Martins, and Andreia Gentil Bonfante. Automated Patent Classification Using Word Embedding. In *Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA '17)*, pages 408–411. IEEE, 2017.

Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, pages 855–864. ACM, 2016.

Bernal Jimenez Gutierrez, Jucheng Zeng, Dongdong Zhang, Ping Zhang, and Yu Su. Document Classification for COVID-19 Literature. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3715–3722. ACL, 2020.

Jacques Guyot, Karim Benzineb, and Gilles Falquet. myClass: A Mature Tool for Patent Classification. In *Proceedings of the International Conference of the Cross-Language Evaluation Forum (CLEF'10)*. CEUR-WS.org, 2010.

Laurel L. Haak, Martin Fenner, Laura Paglione, Ed Pentz, and Howard Ratner. ORCID: A System to Uniquely Identify Researchers. *Learned Publishing*, 25(4):259–264, 2012.

Jason Hepburn. Universal Language Model Fine-tuning for Patent Classification. In *Proceedings of the Australasian Language Technology Association Workshop (ALTA '18)*, pages 93–96. ACL, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

Sebastian Hofstätter, Navid Rekabsaz, Mihai Lupu, Carsten Eickhoff, and Allan Hanbury. Enriching Word Embeddings for Patent Retrieval with Global Context. In *Proceedings of the 41st European Conference on Advances in Information Retrieval (ECIR '19)*, pages 810–818. Springer, 2019.

Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers (ACL '2018)*, pages 328–339. ACL, 2018.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. *CoRR*, abs/2106.09685, 2021.

Jie Hu, Shaobo Li, Jianjun Hu, and Guanci Yang. A Hierarchical Feature Extraction Model for Multi-Label Mechanical Patent Classification. *Sustainability*, 10:1–22, 2018.

Wei Huang, Enhong Chen, Qi Liu, Yuying Chen, Zai Huang, Yang Liu, Zhou Zhao, Dan Zhang, and Shijin Wang. Hierarchical Multi-label Text Classification: An Attention-based Recurrent Network Approach. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*, pages 1051–1060. ACM, 2019.

Wei Huang, Chen Liu, Bo Xiao, Yihua Zhao, Zhaoming Pan, Zhimin Zhang, Xinyun Yang, and Guiquan Liu. Exploring Label Hierarchy in a Generative Way for Hierarchical Text

Classification. In *Proceedings of the 29th International Conference on Computational Linguistics (COLING '22)*, pages 1116–1127. ACL, 2022.

Anette Hulth and Beáta Megyesi. A Study on Automatically Extracted Keywords in Text Categorization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL '06)*. ACL, 2006.

Takashi Inaba and Mariagrazia Squicciarini. ICT: A New Taxonomy Based on the International Patent Classification. *OECD Science, Technology and Industry Working Papers*, 2017.

Peter Ingwersen. Polyrepresentation of Information Needs and Semantic Entities: Elements of a Cognitive Theory for Information Retrieval Interaction. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94)*, pages 101–110. ACM/Springer, 1994.

Mohit Iyyer, Varun Manjunatha, Jordan L. Boyd-Graber, and Hal Daumé III. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP '15), Volume 1: Long Papers*, pages 1681–1691. ACL, 2015.

Ting Jiang, Deqing Wang, Leilei Sun, Zhongzhi Chen, Fuzhen Zhuang, and Qinghong Yang. Exploiting Global and Local Hierarchies for Hierarchical Text Classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP '22)*, pages 4030–4039. ACL, 2022.

Xiao-yu Jiang, Xiao-Zhong Fan, Zhi-Fei Wang, and Ke-Liang Jia. Improving the Performance of Text Categorization Using Automatic Summarization. In *Proceedings of the 2009 International Conference on Computer Modeling and Simulation (ICCMS '09)*, page 347–351. IEEE Computer Society, 2009.

Alistair Johnson, Tom Pollard, Lu Shen, Li-wei Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Celi, and Roger Mark. MIMIC-III, a Freely Accessible Critical Care Database. *Scientific Data*, 3:160035, 2016.

Akanksha Joshi, Eduardo Fidalgo, Enrique Alegre, and Rocío Alaíz-Rodríguez. RankSum - An Unsupervised Extractive Text Summarization Based on Rank Fusion. *Expert Systems with Applications*, 200:116846, 2022.

Dylan Myungchul Kang, Charles Cheolgi Lee, Suan Lee, and Wookey Lee. Patent Prior Art Search Using Deep Learning Language Model. In *Proceedings of the 24th Symposium on International Database Engineering and Applications (IDEAS '20)*, pages 1–5. ACM, 2020.

Sue J. Ker and Jen-Nan Chen. A Text Categorization Based on Summarization Technique. In *Proceedings of the ACL-2000 Workshop on Recent Advances in Natural Language Processing and Information Retrieval: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics*, page 79–83. ACL, 2000.

Yoon Kim. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP '14)*, pages 1746–1751. ACL, 2014.

Gary D. Kimura and Alan C. Shaw. The Structure of Abstract Document Objects. In *Proceedings of the Second ACM-SIGOA Conference on Office Information Systems (COCS '84)*, pages 161–169. ACM, 1984.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR '17)*. OpenReview.net, 2017.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The Efficient Transformer. In *Proceedings of the 8th International Conference on Learning Representations (ICLR '20)*. OpenReview.net, 2020.

Youngjoong Ko, Jinwoo Park, and Jungyun Seo. Automatic Text Categorization using the Importance of Sentences. In *Proceedings of the 19th International Conference on Computational linguistics (COLING '02)*. ACL, 2002.

Aleksander Kolcz, Vidya Prabakarmurthi, and Jugal Kalita. Summarization as Feature Selection for Text Categorization. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM '01)*, page 365–370. ACM, 2001.

Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes. HDLTex: Hierarchical Deep Learning for Text Classification. In *Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA '17)*, pages 364–371. IEEE, 2017.

Ralf Krestel, Renukswamy Chikkamath, Christoph Hewel, and Julian Risch. A Survey on Deep Learning for Patent Analysis. *World Patent Information*, 65:102035, 2021.

Marc Krier and Francesco Zaccà. Automatic Categorisation Applications at the European Patent Office. *World Patent Information*, 24(3):187–196, 2002.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS '12)*, pages 1106–1114. Curran Associates Inc., 2012.

Leah S. Larkey. Some Issues in the Automatic Classification of U.S. Patents. In *Working Notes for the AAAI-98 Workshop on Learning for Text Categorization*, 1998.

Leah S. Larkey. A Patent Search and Classification System. In *Proceedings of the 4th ACM Conference on Digital Libraries (DL '99)*, pages 179–187. ACM, 1999.

Changyong Lee and Gyumin Lee. Technology Opportunity Analysis Based on Recombinant Search: Patent Landscape Analysis for Generation. *Scientometrics*, 121(2):603–632, 2019.

Jieh-Sheng Lee and Jieh Hsiang. PatentBERT: Patent Classification with Fine-Tuning a Pre-Trained BERT Model. *CoRR*, abs/1906.02124, 2019.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: A Pre-Trained Biomedical Language Representation Model for Biomedical Text Mining. *Bioinformatics*, 36(4):1234–1240, 2020.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A New Benchmark Collection for Text Categorization Research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

Huahang Li, Shuangyin Li, Yuncheng Jiang, and Gansen Zhao. CoPatE: A Novel Contrastive Learning Framework for Patent Embeddings. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM '22)*, page 1104–1113. ACM, 2022.

Shaobo Li, Jie Hu, Yuxin Cui, and Jianjun Hu. DeepPatent: Patent Classification with Convolutional Neural Networks and Word Embedding. *Scientometrics*, 117(2):721–744, 2018a.

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. A Unified Syntax-aware Framework for Semantic Role Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP '18)*, pages 2401–2411. ACL, 2018b.

Sora Lim and Yongjin Kwon. IPC Multi-label Classification Based on the Field Functionality of Patent Documents. In *Proceedings of the 12th International Conference on Advanced Data Mining and Applications (ADMA '16)*, pages 677–691. Springer, 2016.

Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics, 2004.

Naiyin Liu, Qianlong Wang, and Jiangtao Ren. Label-Embedding Bi-directional Attentive Model for Multi-label Text Classification. *Neural Processing Letters*, 53(1):375–389, 2021.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, abs/1907.11692, 2019.

Mihai Lupu, Atsushi Fujii, Douglas W. Oard, Makoto Iwayama, and Noriko Kando. *Current Challenges in Patent Information Retrieval*, chapter Patent-Related Tasks at NTCIR, pages 77–111. Springer, 2017.

Xiangru Lyu, Xueqiang Lyu, Fanshu Sun, and Zhian Dong. Patent Domain Terminology Extraction Based on Multi-Feature Fusion and BiLSTM-CRF Model. In *Proceedings of the 4th conference on Fuzzy Systems and Data Mining (FSDM '18)*, volume 309 of *Frontiers in Artificial Intelligence and Applications*, pages 495–500. IOS Press, 2018.

Wolfgang Maass and Veda C. Storey. Pairing Conceptual Modeling with Machine Learning. *Data & Knowledge Engineering*, 134:101909, 2021.

Walid Magdy and Gareth J. F. Jones. A Study on Query Expansion Methods for Patent Retrieval. In *Proceedings of the 4th Workshop on Patent Information Retrieval (PaIR '11)*, pages 19–24. ACM, 2011.

T.M.I. Mahlia, Z.A.H.S. Syazmi, M. Mofijur, A.E. Pg Abas, M.R. Bilad, Hwai Chyuan Ong, and A.S. Silitonga. Patent Landscape Review on Biodiesel Production: Technology Updates. *Renewable and Sustainable Energy Reviews*, 118:109526, 2020.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial Autoencoders. *CoRR*, abs/1511.05644, 2016.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

Alan Marco, Joshua Sarnoff, and Charles deGrazia. Patent Claims and Patent Scope. *Research Policy*, 48:103790, 2019.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. Weakly-Supervised Hierarchical Text Classification. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence (AAAI-IAAI-EAAI '19)*, pages 6826–6833. AAAI Press, 2019.

Rada Mihalcea and Samer Hassan. Using the Essence of Texts to Improve Document Classification. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2005*. INCOMA Ltd., 2005.

Rada Mihalcea and Paul Tarau. TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP '04)*, pages 404–411. ACL, 2004.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the 1st International Conference on Learning Representations (ICLR '13)*, 2013a.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS '13)*, pages 3111–3119. Curran Associates Inc., 2013b.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep Learning–Based Text Classification: A Comprehensive Review. *ACM Computing Surveys*, 54(3), 2021.

Diego Mollá and Dilesha Seneviratne. Overview of the 2018 ALTA Shared Task: Classifying Patent Applications. In *Proceedings of the Australasian Language Technology Association Workshop (ALTA '18)*, pages 84–88, 2018.

Jan Morbach, Ri Hai, Birgit Bayer, and Wolfgang Marquardt. Document Models. In *Collaborative and Distributed Chemical Engineering. From Understanding to Substantial Design Process Support - Results of the IMPROVE Project*, volume 4970 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2008.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. Explainable Prediction of Medical Codes from Clinical Text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '18), Volume 1 (Long Papers)*, pages 1101–1111. ACL, 2018.

Soroosh Nalchigar and Eric Yu. Designing Business Analytics Solutions. *Business and Information Systems Engineering*, 62(1):61–75, 2020.

Soroosh Nalchigar, Eric Yu, and Karim Keshavjee. Modeling Machine Learning Requirements From Three Perspectives: A Case Report from the Healthcare Domain. *Requirements Engineering*, 26(2):237–254, 2021.

Andrew Ng. *CS229 Lecture Notes*. Stanford, 2022.

Muyao Niu and Jie Cai. A Label Informative Wide & Deep Classifier for Patents and Papers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP '19)*, pages 3436–3441. ACL, 2019.

Sean Nordquist and Adam Meyers. On Breadth Alone: Improving the Precision of Terminology Extraction Systems on Patent Corpora. In *Proceedings of the Natural Legal Language Processing Workshop 2022*, pages 1–11. ACL, 2022.

R.S. Norhasyima and T.M.I. Mahlia. Advances in CO_2 Utilization Technology: A Patent Landscape Review. *Journal of CO2 Utilization*, 26:323–335, 2018.

Juri Opitz and Sebastian Burst. Macro F1 and Macro F1. *CoRR*, abs/1911.03347, 2019.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training Language Models to Follow Instructions with Human Feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22)*, volume 35, pages 27730–27744. Curran Associates Inc., 2022.

Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. Hierarchical Transformers for Long Document Classification. In *Proceedings of the 2019 IEEE Workshop on Automatic Speech Recognition and Understanding Workshop (ASRU '19)*, pages 838–844. IEEE, 2019.

Hyunji Hayley Park, Yogarshi Vyas, and Kashif Shah. Efficient Classification of Long Documents Using Transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, Volume 2: Short Papers (ACL '22)*, pages 702–709. ACL, 2022.

Raymond Pearl. Introduction to Medical Biometry and Statistics. *Nature*, 113:563–564, 1924.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web (WWW '18)*, pages 1063–1072. ACM, 2018.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP '14)*, pages 1532–1543. ACL, 2014.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD '14)*, pages 701–710. ACM, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '18), Volume 1 (Long Papers)*, pages 2227–2237. ACL, 2018.

Maxime Peyrard. A Simple Theoretical Model of Importance for Summarization. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL '19), Volume 1: Long Papers*, pages 1059–1073. ACL, 2019.

Virginia Picanço-Castro, Cristiano Gonçalves Pereira, Dimas Tadeu Covas, Geciane Silveira Porto, Aglaia Athanassiadou, and Marxa Leão Figueiredo. Emerging Patent Landscape for Non-Viral Vectors Used for Gene Therapy. *Nature Biotechnology*, 38(2):151–157, 2020.

Florina Piroi and Allan Hanbury. *Evaluating Information Retrieval Systems on European Patent Data: The CLEF-IP Campaign*, pages 113–142. Springer, 2017.

Florina Piroi, Mihai Lupu, Allan Hanbury, and Veronika Zenz. CLEF-IP 2011: Retrieval in the Intellectual Property Domain. In *CLEF 2011 Labs and Workshop, Notebook Papers*, volume 1177 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.

Ekta Priyadarshini, G. Raj Gayathri, Samuel E Chakravarthy, Mohanakrishnan Vidhya, and W.Abitha Memala. Analysis of Heart Disease Using Statistical Techniques. *Journal of Physics: Conference Series*, 1770, 2021.

Xiao Pu, Mingqi Gao, and Xiaojun Wan. Summarization is (Almost) Dead. *CoRR*, abs/2309.09558, 2023.

Subhash Chandra Pujari, Annemarie Friedrich, and Jannik Strötgen. A Multi-task Approach to Neural Multi-label Hierarchical Patent Classification Using Transformers. In *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part I*, volume 12656 of *Lecture Notes in Computer Science*, pages 513–528. Springer, 2021a.

Subhash Chandra Pujari, Tim Tarsi, Jannik Strötgen, and Annemarie Friedrich. Explainable Prediction of Medical Codes from Clinical Text. In *Proceedings of the BioCreative VII Challenge Evaluation Workshop*, 2021b.

Subhash Chandra Pujari, Fryderyk Mantiuk, Mark Giereth, Jannik Strötgen, and Annemarie Friedrich. Evaluating Neural Multi-Field Document Representations for Patent Classification. In *Proceedings of the 12th International Workshop on Bibliometric-enhanced Information Retrieval (BIR '22) co-located with 44th European Conference on Information Retrieval (ECIR '22)*, volume 3230 of *CEUR Workshop Proceedings*, pages 13–27. CEUR-WS.org, 2022a.

Subhash Chandra Pujari, Jannik Strötgen, Mark Giereth, Michael Gertz, and Annemarie Friedrich. Three Real-World Datasets and Neural Computational Models for Classification Tasks in Patent Landscaping. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP '22)*, pages 11498–11513. ACL, 2022b.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP '19)*, pages 3982–3992. ACL, 2019.

Steffen Remus, Rami Aly, and Chris Biemann. GermEval 2019 Task 1: Hierarchical Classification of Blurbs. In *Proceedings of the 15th Conference on Natural Language Processing, (KONVENS '19)*, pages 280–292. German Society for Computational Linguistics & Language Technology, 2019.

Georg Richter and Andrew MacFarlane. The Impact of Metadata on the Accuracy of Automated Patent Classification. *World Patent Information*, 27(1):13–26, 2005.

Julian Risch and Ralf Krestel. Learning Patent Speak: Investigating Domain-Specific Word Embeddings. In *Proceedings of the 13th IEEE International Conference on Digital Information Management (ICDIM '18)*, pages 63–68. IEEE, 2018.

Julian Risch and Ralf Krestel. Domain-Specific Word Embeddings for Patent Classification. *Data Technologies and Applications*, 53(1):108–122, 2019.

Julian Risch, Nicolas Alder, Christoph Hewel, and Ralf Krestel. PatentMatch: A Dataset for Matching Patent Claims & Prior Art. *CoRR*, abs/2012.13919, 2020a.

Julian Risch, Samuele Garda, and Ralf Krestel. Hierarchical Document Classification as a Sequence Generation Task. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL '20)*, pages 147–155. ACM/IEEE, 2020b.

Kervy Rivas Rojas, Gina Bustamante, Arturo Oncevay, and Marco Antonio Sobrevilla Cabezudo. Efficient Strategies for Hierarchical Text Classification: External Knowledge and Auxiliary Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL '20)*, pages 2252–2257. ACL, 2020.

Frank Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological review*, 65 6:386–408, 1958.

Arousha Haghighian Roudsari, Jafar Afshar, Wookey Lee, and Suan Lee. PatentNet: Multi-Label Classification of Patent Documents using Deep Learning based Language Understanding. *Scientometrics*, 127(1):207–231, 2022.

Benedek Rozemberczki and Rik Sarkar. Fast Sequence-Based Embedding with Diffusion Graphs. In *Complex Networks IX*, pages 99–107. Springer, 2018.

Sebastian Ruder. An Overview of Multi-Task Learning in Deep Neural Networks. *CoRR*, abs/1706.05098, 2017.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic Routing between Capsules. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS '17)*, page 3859–3869. Curran Associates Inc., 2017.

Mobashir Sadat and Cornelia Caragea. Hierarchical Multi-Label Classification of Scientific Documents. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP '22)*, pages 8923–8937. ACL, 2022.

Mike Schuster and Kuldip K. Paliwal. Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

Marawan Shalaby, Jan Stutzki, Matthias Schubert, and Stephan Günnemann. An LSTM Approach to Patent Classification based on Fixed Hierarchy Vectors. In *Proceedings of the 2018 SIAM International Conference on Data Mining (SDM '18)*, pages 495–503. SIAM, 2018.

Walid Shalaby and Wlodek Zadrozny. Patent Retrieval: A Literature Review. *Knowledge and Information Systems*, 61(2):631–660, 2019.

Eva Sharma, Chen Li, and Lu Wang. BIGPATENT: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL '19), Volume 1: Long Papers*, pages 2204–2213. ACL, 2019.

Jiaming Shen, Wenda Qiu, Yu Meng, Jingbo Shang, Xiang Ren, and Jiawei Han. Taxo-Class: Hierarchical Multi-Label Text Classification Using Only Class Names. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '21)*, pages 4239–4249. ACL, 2021.

Kazuya Shimura, Jiyi Li, and Fumiyo Fukumoto. HFT-CNN: Learning Hierarchical Category Structure for Multi-label Short Text Categorization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP '18)*, pages 811–816. ACL, 2018.

Carlos N. Silla and Alex A. Freitas. A Survey of Hierarchical Classification across Different Application Domains. *Data Mining Knowledge Discovery*, 22(1–2):31–72, jan 2011. ISSN 1384-5810.

Karen Sparck Jones. *A Statistical Interpretation of Term Specificity and Its Application in Retrieval*, page 132–142. Taylor Graham Publishing, GBR, 1988.

Andreas Spitz, Dennis Aumiller, Bálint Soproni, and Michael Gertz. A Versatile Hypergraph Model for Document Collections. In *Proceedings of the 32nd International Conference on Scientific and Statistical Database Management (SSDBM '20)*, pages 7:1–7:12. ACM, 2020.

Chul Sung, Tejas Dhamecha, Swarnadeep Saha, Tengfei Ma, Vinay Reddy, and Rishi Arora. Pre-Training BERT on Domain Resources for Short Answer Grading. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP '19)*, pages 6071–6075. ACL, 2019.

Mirac Suzgun, Luke Melas-Kyriazi, Suproteem K. Sarkar, Scott Duke Kominers, and Stuart M. Shieber. The Harvard USPTO Patent Dataset: A Large-Scale, Well-Structured, and Multi-Purpose Corpus of Patent Applications. *CoRR*, abs/2207.04043, 2022.

Pingjie Tang, Meng Jiang, Bryan (Ning) Xia, Jed W. Pitera, Jeffrey Welser, and Nitesh V. Chawla. Multi-Label Patent Categorization with Non-Local Attention-Based Graph Convolutional Network. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*, pages 9024–9031, 2020.

Andrew A. Toole, Nicholas A. Pairolero, Alexander V. Giczy, James Q. Forman, Jesse Frumkin, David B. Orange, Anne Thomas Homescu, Steve Melnick, Christyann Pulliam, Matthew Such, Kakali Chaki, Eric Nilsson, Ying Yu Chen, Vincent M. Gonzales, Ben M. Rifkin, and Christian Hannon. Inventing AI: Tracing the Diffusion of Artificial Intelligence with U.S. Patents. Technical report, United States Patent and Trademark Office, 2020.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models. *CoRR*, abs/2302.13971, 2023.

Tung Tran and Ramakanth Kavuluru. Supervised Approaches to Assign Cooperative Patent Classification (CPC) Codes to Patents. In *Proceedings of the 5th International Conference on Mining Intelligence and Knowledge Exploration (MIKE '17)*, pages 22–34. Springer, 2017.

J Truett, Jerome Cornfield, and William B. Kannel. A Multivariate Analysis of the Risk of Coronary Heart Disease in Framingham. *Journal of Chronic Diseases*, 20 7:511–24, 1967.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS '17)*, pages 5998–6008. Curran Associates Inc., 2017.

Suzan Verberne and Eva D'hondt. Patent Classification Experiments with the Linguistic Classification System LCS in CLEF-IP 2011. In *Proceedings of the International Conference of the Cross-Language Evaluation Forum (CLEF'11)*. CEUR-WS.org, 2011.

Mihai Vlase, Dan Munteanu, and Adrian Istrate. Improvement of K-Means Clustering Using Patents Metadata. In *Proceedings of the 8th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM '12)*, pages 293–305. Springer, 2012.

Thanh Vu, Dat Quoc Nguyen, and Anthony N. Nguyen. A Label Attention Model for ICD Coding from Clinical Text. In Christian Bessiere, editor, *Proceedings of the 29th*

International Joint Conference on Artificial Intelligence (IJCAI '20), pages 3335–3341. International Joint Conferences on Artificial Intelligence Organization, 2020.

Xuepeng Wang, Li Zhao, Bing Liu, Tao Chen, Feng Zhang, and Di Wang. Concept-Based Label Embedding via Dynamic Routing for Hierarchical Text Classification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Volume 1: Long Papers (ACL-IJCNLP '21)*, pages 5010–5019. ACL, 2021.

Yue Wang, Dan Qiao, Juntao Li, Jinxiong Chang, Qishen Zhang, Zhongyi Liu, Guannan Zhang, and Min Zhang. Towards Better Hierarchical Text Classification with Data Generation. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7722–7739. ACL, 2023.

Zihan Wang, Peiyi Wang, Lianzhe Huang, Xin Sun, and Houfeng Wang. Incorporating Hierarchy into Text Encoder: a Contrastive Learning Approach for Hierarchical Text Classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL '22)*, pages 7109–7119. ACL, 2022a.

Zihan Wang, Peiyi Wang, Tianyu Liu, Binghuai Lin, Yunbo Cao, Zhifang Sui, and Houfeng Wang. HPT: Hierarchy-aware Prompt Tuning for Hierarchical Text Classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP '22)*, pages 3740–3751. ACL, 2022b.

Jonatas Wehrmann, Ricardo Cerri, and Rodrigo C. Barros. Hierarchical Multi-Label Classification Networks. In *Proceedings of the 35th International Conference on Machine Learning (ICML '18)*, pages 5225–5234. PMLR, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR*, abs/1910.03771, 2019.

Chih-Hung Wu, Yun Ken, and Tao Huang. Patent Classification System Using a New Hybrid Genetic Algorithm Support Vector Machine. *Applied Soft Computing*, 10(4): 1164–1177, 2010.

Lin Xiao, Xin Huang, Boli Chen, and Liping Jing. Label-Specific Document Representation for Multi-Label Text Classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP '19)*, pages 466–475. ACL, 2019.

Yijin Xiong, Yukun Feng, Hao Wu, Hidetaka Kamigaito, and Manabu Okumura. Fusing Label Embedding into BERT: An Efficient Improvement for Text Classification. In *Findings of the Association for Computational Linguistics (ACL-IJCNLP '21)*. ACL, 2021.

Linli Xu, Sijie Teng, Ruoyu Zhao, Junliang Guo, Chi Xiao, Deqiang Jiang, and Bo Ren. Hierarchical Multi-label Text Classification with Horizontal and Vertical Category Correlations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP '21)*, pages 2459–2468. ACL, 2021.

Xiaobing Xue and W. Bruce Croft. Automatic Query Generation for Patent Search. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*, pages 2037–2040. ACM, 2009.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *CoRR*, abs/1906.08237, 2020.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. Hierarchical Attention Networks for Document Classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '16)*, pages 1480–1489. ACL, 2016.

Liang Yao, Chengsheng Mao, and Yuan Luo. Graph Convolutional Networks for Text Classification. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence and 31st Innovative Applications of Artificial Intelligence Conference and 9th AAAI Symposium on Educational Advances in Artificial Intelligence (AAAI-IAAI-EAAI '19)*, pages 7370–7377. AAAI Press, 2019.

Selen Yücesoy Kahraman, Türkay Dereli, and Alptekin Durmuşoğlu. Forty Years of Automated Patent Classification. *International Journal of Information Technology & Decision Making*, pages 1–32, 2023.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for Longer Sequences. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20)*. Curran Associates Inc., 2020.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into Deep Learning. *CoRR*, abs/2106.11342, 2021.

Ruohong Zhang, Yau-Shian Wang, Yiming Yang, Donghan Yu, Tom Vu, and Likun Lei. Long-tailed Extreme Multi-label Text Classification by the Retrieval of Generated Pseudo Label Descriptions. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1092–1106. ACL, 2023.

Hao Zheng and Mirella Lapata. Sentence Centrality Revisited for Unsupervised Summarization. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL '19), Volume 1: Long Papers*, pages 6236–6247. ACL, 2019.

Jie Zhou, Chunping Ma, Dingkun Long, Guangwei Xu, Ning Ding, Haoyu Zhang, Pengjun Xie, and Gongshen Liu. Hierarchy-Aware Global Model for Hierarchical Text Classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL '20)*, pages 1106–1117. ACL, 2020.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING '16)*, pages 3485–3495. ACL, 2016.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV '15)*, pages 19–27. IEEE, 2015.

Geoffrey Zweig and Chris J.C. Burges. The Microsoft Research Sentence Completion Challenge. Technical Report MSR-TR-2011-129, Microsoft Research, December 2011. URL https://www.microsoft.com/en-us/research/publication/the-microsoft-research-sentence-completion-challenge/.