University of Passau
Faculty of Computer Science and Mathematics
Chair for Data & Knowledge Engineering

Master Thesis in Informatics

Ill-formedness Analysis and Refinement of Natural Language Queries for Prior Art Search Engines

submitted by

Deepak Rastogi

1. Examiner: Prof. Dr. Markus Endres
2. Examiner: Prof. Dr. Michael Granitzer
Supervisor: Renukswamy Chikkamath
Date: March 24, 2023

Contents

Abstract

In this thesis, we assess the effectiveness of natural language queries aimed to search prior art in the patent domain using an AI based prior art search engine (like PQAI). We identify deficiencies in articulation of queries using three different state-of-the-art models and obtain a score that points out the level of quality of a query. We train a grammatical error detection model using a pretrained BERT model (Bert-for-patents) on a new dataset of patent abstracts with five types of grammatical errors. This model achieves a whopping 14% increase in F1 score over the baseline. The second model, called Query Specificity Model (QSM), is trained to annotate each sentence in patent queries as general or specific. The QSM works on a hypothesis that a natural language query should talk more specifically about the invention rather than talking about general facts. This model outperforms the current state-of-the-art model by an increase of 8.25% in accuracy. The third, Query Verbosity Model (QVM), predicts the actual length of a patent abstract when given a tiny 50-word snippet from the same abstract. QVM helps find if a query is verbose (contains unnecessary information) or if a query is too short of describing an invention. QVM is a two stage model that performs both classification and regression over 87 semantic and syntactic properties capturing insights of each text snippet. The classification model is an XGBoost model that predicts the range of lengths of a snippet. The Ordinal Least Squares (OLS) linear regression model predicts the actual lengths of the abstract. Based on the length predicted by this model, we compare the results with its actual length and determine the verbosity degree in a text. Our QVM length range prediction model shows a 6% increase in accuracy than the state-of-the-art model. The length range prediction model performs almost similarly to the state-of-the-art model. The last step in a model is to check how relevant the results provided by PQAI are when given a query.

Acknowledgments

First and foremost, I would like to express my heartfelt gratitude to Prof. Dr. Markus Endres for providing me with a master's thesis and allowing me to work on such a fascinating topic.

I would like to sincerely thank Prof. Dr. Michael Granitzer for agreeing to become the second examiner of my thesis.

My thesis advisor, Renukswamy Chikkamath, has my sincere gratitude for his assistance with this thesis. I believe there could not have been a better thesis advisor than him. Without his constant feedback, this would not have been able to be finished successfully. He gave me absolute freedom from choosing a thesis topic to how I wanted to conduct my experiments. This helped me approach the subject broadly and carry out my tests effectively.

I sincerely thank Mr. Mahesh Maan (Developer of PQAI) for serving as my thesis' second advisor. I have to admire his humility and willingness to provide help whenever necessary. Despite being in a different time zone, he was always accessible for meetings at German times. He was also my first point of contact whenever I had any concerns, and he always responded quickly.

Additionally, I would like to thank Vishvapal Sinhji Parmar (Research Assistant at the University of Passau); Niharika Kumari (Data Scientist at Renolit SE, Worms), Abbu Bakkar Siddiqui (Software Engineer, Dubai police, Dubai), Harish Krishna (Sales and Marketing Manager, Toyota, India) and Harish Rao (Software Engineer, Cognizant services, India) for participating in the annotation survey.

Finally, I must express my sincere thanks to my parents and to every member of my family for their unwavering support and never-ending encouragement during my years of study as well as during the process of conducting research for and writing this thesis. Without them, this accomplishment would not have been possible. Thank You.

Deepak Rastogi

List of Figures

List of Tables

1 Introduction

1.1 Introduction and Motivation

A patent is a type of intellectual property that provides legal rights to an inventor to prevent others from making, using, or selling an invention without the proper consent of the owner. These patents are filed with all necessary technical details to reproduce the invention. However, an inventor needs to know that his/her invention is unique and not already known. Therefore, to file a patent application initially, a brief search of patent documents, research journals, and product manuals that could be a barrier to the invention is carried out before filing a patent.

Prior art is the information that has been disclosed to the public before filing the patent application. The source of this information may include granted patents, scientific journals, and books, etc. In some cases, other forms, such as unpublished patents, historical knowledge, and even prehistoric knowledge, can be treated as prior art. The goal of the prior art search is to determine the novelty of an invention. However, prior art search is intensively time-consuming and tedious work because an inventor (or an examiner) has to go through much information known in the public domain. Therefore, prior art search an interesting and practical problem for Natural Language Processing / Understanding (NLP/NLU) researchers. There are numerous search engines (E.g., Google Patents) available to search for prior arts. However, these search engines may provide hundreds (sometimes thousands) of search results, but most of them might not be relevant for an invention. Those results are there only because some keywords of the query match the search results. Unlike other search engines, PQAI is an AI-based prior art search platform that help inventors narrow their prior art search. PQAI requires a well-written natural language query specifying essential detail of an invention to get highly relevant results. It is evident that if a query varies, obtained results also vary. Hence, a query has to be well-written and well-formatted.

PQAI (Patent Quality through Artificial Intelligence) system is an initiative by ATT and Georgia Intellectual Property Alliance (GIPA). It is an open-source prior art search tool developed with the idea of searching for the most relevant prior art. In traditional prior art search, a user forms a query, feeds it to the search engine, and returns hundreds and thousands of results based on keyword matches. The results that come at the top are the most relevant. However, some results may not appear due to variations in vocabulary. So the user of such a search engine has to invest time to form queries by changing some vocabulary repeatedly. There are very few among these resultant documents that are relevant, and the rest of them are often irrelevant pieces of information. The user spends most of his time reading the documents, which are eventually discarded. Therefore, a search engine must judge the irrelevance of a document. Unfortunately, most prior art search engines put less emphasis on this part. PQAI, on the other hand, enables inventors to judge the relevance of documents

quickly. It goes one step beyond by picking out the relevant parts of the documents matching queries called "snippets" or "passages ."They are complete sentences or parts of sentences that make sense independently. Thus, the inventor spends much less time sifting through irrelevant results. PQAI is not limited to just patents but gives results, including articles, research papers, and more. PQAI database has 11 million US patents and applications and around 11.5 million research papers.

A query for a prior art search engine is different from a query for a generic search engine, e.g., Google. Usually, generic search engine queries combine a few keywords or phrases. Such queries typically target a user's information need. E.g. "medicine for fever and headache" is a prevalent topic and thousands of articles are available on the web addressing this particular query. Hence, a well-optimized search engine provides results on medication that treats fever and headache together. A search engine that shows the first result as "medicines for fever" and the second as "medicine for headache" is not most effective for this query. Now, suppose we want to introduce more constraints in this query, like the specifics of the patient, other minor symptoms, and information about recent surgery or disease. With such information, the query becomes specialized, making it difficult to parse and present relevant results. It may be possible that search engines run out of results satisfying all information needs. Generally, these search engines are not built to process such a big query with so much information.

On the other hand, prior art search queries are bound to have more information, describing many specific details of one's invention. However, an invention without specifics may be difficult to comprehend, even for humans. A well-formed natural language prior art search query should comprise sufficient details. It should be in a form that allows the search system to determine prior inventions conceptually similar to the invention behind the search query. Not all users may form their queries well. There is a possibility that users of two continents use different keywords for the same thing. E.g., mobile phones and cellphones both indicate the same meaning. Moreover, the writing style of a sentence varies from person to person, depending on their personality or geographic location. E.g., Somebody in the USA might say, "The car has run out of gas," whereas someone from Asia would say, "There is no petrol in the car." Technically, "gas" has a different meaning, but they both refer to "fuel" in this context. The language proficiency of the user also plays an important role. E.g., an inventor from China may have limited vocabulary knowledge compared to a person from the United States. As a result, a user may end up writing sub-optimal queries.

Some characteristics of well-formed prior art search queries are the following.

1. A well-formed prior art search query should be unambiguously parsable to reveal the underlying interrelationships among its constituent concepts. Therefore, it should be grammatically accurate.
2. A well-formed prior art search query should also preferably use terms with concrete and specific meanings so as not to be open to too many interpretations.
3. A well-formed query should not contain irrelevant or redundant information.

1.2 Research Questions

In the real world, not all users formulate queries well. Some users may create queries that are too short of describing the invention. For example, the query "adjustable mechanical keyboard" does not say enough to be very useful for running a prior art search. Other users may create overly verbose queries that focus too much on the non-technical aspects of a technical idea thus reduce the signal-to-noise ratio in the query.

Think of descriptions with very elaborate background art. Other queries may be overly broad and open to interpretation. Typically this happens when the claim language itself is used as a query. In claims, a pen, for instance, can be described as a "marking device" - which is hardly a good term for a prior art search system. Most of these issues can be resolved by providing the user feedback about whether the query is good or not and asking the user to re-articulate it if required. However, the current search engine still expects some technical advancements. One of them is query suggestions. Currently, this search engine lacks predictive text and query quality scores. Query quality scorer is a basic need of all sorts of search engines, whether a standard web search engine or a task-specific search engine. It helps the user reanalyze his query using specific keywords and rephrasing.

As we discussed in the previous chapter, getting relevant results from a search engine depends not only on the technical capability of the search engine but also on the sense of forming a good query. However, one may ask the following questions:

1. **What are the differences between patent and generic search engine queries?** A generic search engine and patent query could differ in various ways. In general, patent queries are simple, plain language queries that include a concise summary of the invention. These queries are typically 50–100 words long. Generic search engine queries, on the other hand, are short and to the point, 10–20 words long. As a result, patent queries contain more information about the information need. In later chapters, we will examine some examples of patent queries.
2. **Why do patent queries have to be well-formed?** A well-formed patent query directly affects the prior art search engine's functionality. Using phrase level matching, generic search engines rank the results based on how many of the query's common keywords and phrases they contain. However, AI based prior art (domain-specific) search engines operate differently compared to general search engines. We will briefly review the design and operation of such a search engine.
3. **What are the possible characteristics of ill-formed queries?** Many number of flaws could exist in a patent search. For instance, the query can have grammatical faults, lack essential details of an invention, leave room for multiple interpretations, or include irrelevant information that may throw the engine off-track. We will explore these characteristics of improperly formatted queries in more detail and suggest ways to spot them.
4. **How do we detect if a query is ill-formed?** A patent search could have several types of flaws. We suggest specialized models in this thesis to find these errors in the query text. Three models for detecting grammatical faults, verbosity, and level of specificity in the question will be covered.

5. **Does removing these flaws from the query improve the quality of results?** It is crucial to understand how a search query with the defects addressed in the thesis affects the quality of the returned results. To determine the extent to which these defects impact the query's performance, we will run queries with discovered faults on the prior art search engine and their corrected counterpart.
6. **Can the proposed methods be used on any text other than prior art search queries?** The issues we will discuss in this thesis are not exclusive to texts in the patent field; they can appear in texts from any field. Therefore, the suggested approaches ought to be reliable. While the skeleton and methodology of our models can be applied to any text data, they are only designed explicitly for the texts of patents. The methods section of this thesis provides a brief explanation of our models' operating principles, which can be used to train models for texts from any domain.

1.3 Structure of the Thesis

In this thesis, we focus on the three types of faults, namely grammatical errors, a lack of specific information, and the existence of irrelevant elements that may appear in a natural language query for a prior art search engine. We resolve these problems by training three models that will enable us to determine whether an NL query has these problems. A complete flow chart of the entire thesis is visualized in Figure 1.1. We shall go into great depth about these models. In the latter sections of this thesis, we examine the PQAI search engine's relevancy of queries with these errors and note how these ill-formedness characteristics affect the standard of search results.

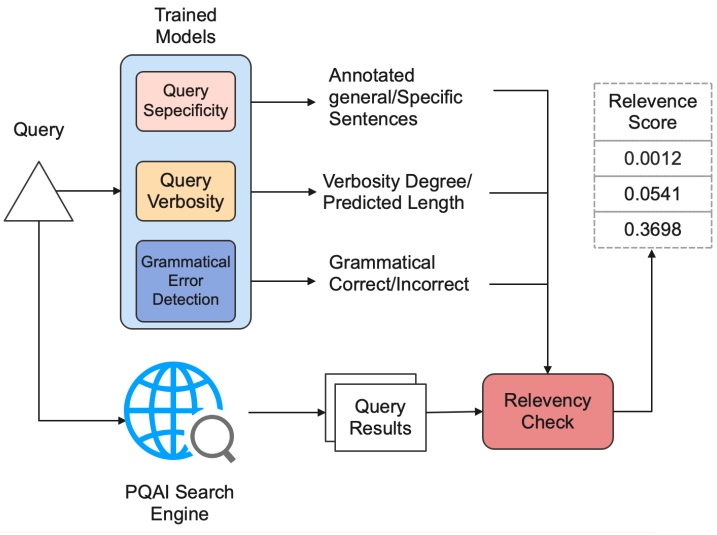

Figure 1.1: Process Flow

Our thesis follows the following structure.

1.3.1 Chapter 2: Related Work

In this chapter, we will cover the earlier research done by other scholars in the fields of prior art search and error detection in regular text. Each connected piece of work, concisely explained in a paragraph, briefly describes the suggested approach and any potential drawbacks to the study.

1.3.2 Chapter 3: Background

The background section is included to inform the reader of the theoretical foundations of various subjects that we expect them to be familiar with before reading the methods section. These subjects will aid readers in comprehending the main idea and objectives of the experiments. Each topic in this section is briefly discussed first, and then its justification and application in our thesis are discussed.

1.3.3 Chapter 4: Grammatical Error Detection (GED)

The methodology part of our thesis starts from this chapter. In this chapter, we train two models, referred to as coarse-grained and fine-grained models, to detect grammatical errors in texts. While the fine-grained GED model predicts certain faults, such as incorrect verb form, article, spelling, etc., the coarse-grained GED model only determines whether a text contains grammatical problems. In the technique section, we will briefly discuss how different grammatical faults are introduced into the text and offer two strategies for spotting them.

1.3.4 Chapter 5: Query Specificity Model (QSM)

In this chapter, we will train a logistic regression (LR) model to determine whether a query sentence provides general information or in-depth details. We will use the Penn Discourse Treebank Dataset [Pra+08] to split down implicit instantiation and specification sentences, treating the first as general and the second as specific [LN11]. To anticipate the annotations, we later train the LR model with an *liblinear* [Fan+08] solver. We used three datasets to train this model. In order, instantiation statements alone, specification sentences alone, and a combination of both in the third.

1.3.5 Chapter 6: Query Verbosity Model (QVM)

In this chapter, a method for finding irrelevant details in a text is called QVM, is introduced. This operates on the principle that a text's content type directly affects the text's length. To predict the lengths of abstracts from the BIGPATENT dataset [SLW19], we use the method by Louis et al. [LN14]. We train a classification model with the extracted features to predict the article length range. Second, we develop an Ordinal least squares (OLS) linear regression model to predict the articles' actual durations. With this method, we train the model by extracting topics from the texts using a newly proposed method. In conclusion, we use PQAI with low and high verbosity levels to assess the results' consistency.

1.3.6 Chapter 7: Results

In this chapter, we talk about how well the models and algorithms from chapters 4, 5, and 6 performed. We give numerous tables and figures to illustrate the outcomes of each suggested model for finding errors in the prior art queries in order to keep things simple.

1.3.7 Chapter 8: Discussion

Our thesis' discussion chapter is a crucial component. In this chapter, we go over how the suggested techniques for identifying errors in queries affect the search engine's general functionality and the relevancy of the returned results. We compare the relevancy of queries having grammatical errors and their correct version. Similarly, we compare the relevancy of a query having a higher verbosity degree than its paraphrased version with a relatively lower verbosity degree. At last, we search using the general sentences, and later general sentences paired with specific sentences predicted by our QSM.

1.3.8 Chapter 9: Conclusion and Future Work

In this chapter, we discuss the justifications for the significance of this research as well as some of its drawbacks. We also talk about possible future studies and trials in this field.

2 Related Work

There is not much work previously done in the domain of natural language prior art search engine queries. It is because most search engines use the traditional approach of finding results for the user queries where the results are ranked based on matching phrases with the user query. Multiple kinds of research are performed to extract relevant phrases from the patent documents and treat them as query terms. Toucedo et al. [TL09] suggested several strategies to extract the most critical terms from the textual sections of the patents.

Although when talking about natural language search queries, some work is available in fields other than the patent domain. Faruqui et al. [FD18] proposed an approach to identify if natural language question on the Google search engine is well-formed. Their work involves a dataset that questions and the well-formedness score annotated by humans. Their definition of a well-formedness of a natural language query is that the query should be grammatical, a straightforward question, and not contain grammatical errors. Although their given definition may only be dealing with grammatical errors, this has inspired us to create a model dealing with various grammatical errors. A dataset with query questions and a well-formedness rating was also made available. However, we would not use their dataset in our thesis for three reasons: first, their queries are formatted as questions, while Our patent queries are concise descriptions of patents. Secondly, their well-formed score is dependent not just on grammatical errors but also on how the questions are constructed, while our model does not deal with queries with the question-like format. The third reason is that their queries are not related to patents. They use a two-layered feed-forward neural network model to predict the well-formedness score of the query. Chu and Zewei et al. [Chu+20] propose a similar approach to converting ill-formed questions to well-formed questions. Their description of a well-formed query is also the same as in Faruqui et al. [FD18].

A book *Toward clarity and grace* [Wil90] is an excellent tool for those working in the natural language text reformulation domain. This book briefly explains many flaws users make that contribute to making a text ill-formed. For example, it briefly discusses verbosity and the types of sentences or terms that may introduce verbosity in a text. It also suggests approaches with examples to identify and rectify those errors in a text apart from grammatical errors.

A natural language query alone is just a piece of text that could be poorly written. Therefore, it would be wise to treat the query as a standard text and identify its flaws. Louis et al. [Lou13] have done excellent work identifying defects in the text. Their approach to detecting specificity and verbosity in the text shows excellent results. However, their work is limited to identifying those flaws only. Their work is purely done using news articles from various sources.

3 Background

This chapter will discuss the technologies used in the further thesis. It includes all the theoretical side of the methods that might be useful to understand our experiments for accessing the quality of a natural language query.

3.1 PQAI

Patent Quality Artificial Intelligence (PQAI) is an artificial intelligence (AI) based search engine that helps inventors and researchers retrieve prior art for their search engine. PQAI searches patents and other technical literature for prior work that is similar to an invention description. It parses the input, locates related prior art, and displays the outcomes using a variety of machine learning (ML) models. PQAI is a modular system whose components can be used individually to enable the research community working in the patent retrieval and data mining domain to easily collaborate and build upon each other's work. The PQAI search engine involves eight component, which is discussed below [MZS21].

- **Patent Database:** Patent Database is abstract component storage that may be made accessible to all other components. The primary benefit of this strategy is that components can circulate references within the database rather than actual patent data.
- **Encoder:** An encoder or Patent Vectorizer takes a patent number as input and produces a vector embedding in a high dimensional space corresponding to the specified patent.
- **Index:** A Patent Vector Index takes a query vector and returns a list of patent numbers that are the best matches for the query.
- **Ranker:** When given a set of patents and user queries, the ranker would sort the patents according to how relevant they are to the query.
- **Classifier:** A patent classifier accepts a list of patent numbers as input and assigns a label. Classifiers can employ customizable classifier models internally, which can be initialized with inputs such as pairs or textual descriptions.
- **Consolidator:** A consolidator accepts a collection of patents and assigns each patent one of a limited number of arbitrary designations. Essentially, it groups patents into clusters where each cluster's patents have specific traits.
- **Filter:** The filter component selects patents that meet a requirement, like publication date.
- **Sorter:** A Patent Sorter takes a list of patent numbers and arranges them so that any patent in the list is followed by the other patents in the list that are closest in similarity.

After receiving a user query input, the PQAI search engine outputs a ranked list of patents, which may contain filters like publication date. Two encoders are employed in the search engine's architecture (Figure 3.1), one for encoding user queries and the other for encoding patents retrieved from the patent database. Based on the requirements of the query, the indexer delivers a list of patents. If the user query contains filtering criteria, the patents are finally filtered. The returned patents are ranked in order of decreasing relevance to the query with the help of a ranker [MZS21].

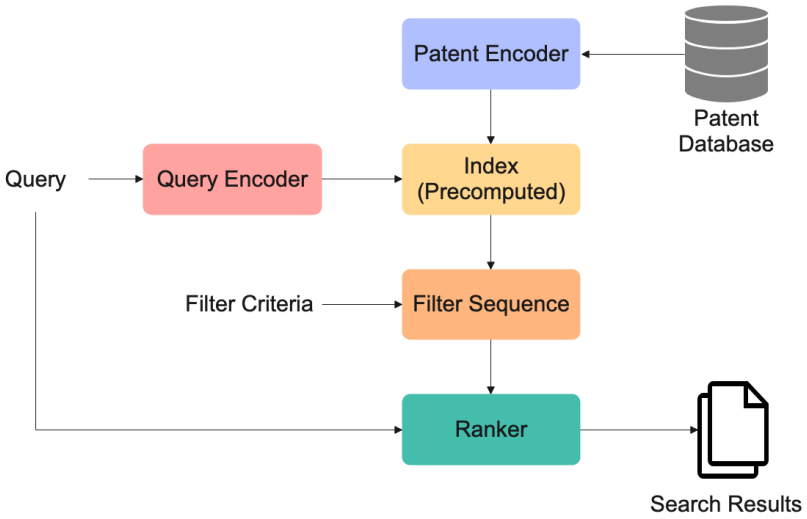

Figure 3.1: Architecture of PQAI

3.2 Transfer Learning

Transfer learning is a machine learning technique in which a model created for one task (called a pretrained model) is utilized as the foundation for a model on another [Bro19]. Pretrained models are typically developed using massive datasets that serve as industry benchmarks. The weights obtained from the models could be applied to further tasks. Transfer learning is very valuable when our training dataset is smaller. In that situation, we could, for instance, initialize the new model weights using the weights from the pretrained models [Mwi21]. Transfer learning generally outperforms traditional machine learning in several ways. It comprises reducing the time required for model training, improving performance, and not requiring a large amount of training data in the target domain. The domain of Natural Language Processing (NLP) has seen the emergence of various transfer learning approaches and architectures over the past few years that have dramatically outperformed the state-of-the-art on various NLP tasks [Rud+19]. Even though the NLP domain has a wide taxonomy of transfer learning, in this thesis, we will be focusing on Sequential Transfer Learning (STL). The aim of STL is to transfer knowledge across a series of processes when the source and target tasks are not always the same. The model is trained on source data in the first pretraining stage, and the intended task is trained on the source model in the second adaptation stage.

3.3 Bidirectional Encoder Representations from Transformers (BERT)

BERT is a natural language processing method built on transformers. The transformer is an attention mechanism that understands the relationships between the context of the words in a sentence. The transformer contains a decoder that creates a task prediction after reading the text input from the encoder [Vas+17]. Only the encoder mechanism is required because BERT aims to produce a language model. The word sequence is read in its entirety by a transformer encoder. This trait enables the model to understand a word's context by considering all of its surroundings. By simply fine-tuning all previously trained parameters, the fine-tuning technique trains on downstream tasks with the fewest possible task-specific parameters [Dev+18]. To predict randomly masked or substituted words, BERT employs a Masked Language Model (visualized in Figure 3.2). BERT is the first fine-tuning-based representation model to produce cutting-edge outcomes for various NLP tasks, highlighting the great potential of the technique [Sun+19]. Before feeding word sequences into the BERT, 15% of each sequence's words are changed to [MASK] tokens. Using the context provided by the other non-masked words in the sequence, the model then tries to guess what the original value of the masked words was. In technical terms, adding a classification layer on top of the encoder output and multiplying the output vectors by the embedding matrix to convert them into the vocabulary dimension are required to predict the output words. The probability of each word in the vocabulary is then calculated using softmax [Sun+19].

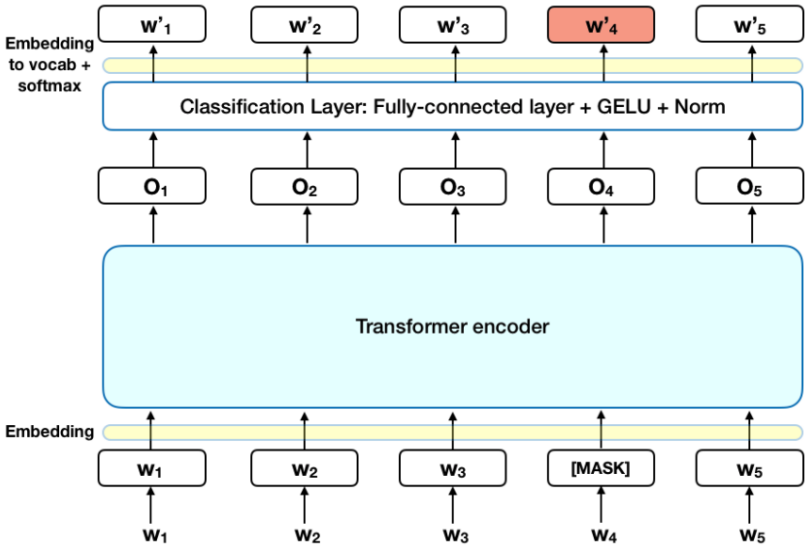

Figure 3.2: Maked LM

We fine-tune a BERT model in our Grammatical Error Detection Model. The task is a text classification model that identifies documents containing various grammatical errors. We fine-tune a pretrained BERT model called BERT-for-Patents[1] to accomplish the task. It is based on the $BERT_{Large}$ model trained using more than 100 million patent documents, including

[1] https://huggingface.co/anferico/bert-for-patents

the sections like abstracts, claims, and descriptions. Based on the $BERT_{Large}$, the BERT-for-Patents model has the same configurations, i.e., the maximum input sequence length is 512 tokens, and the maximum masked words for a sequence is 45. To perform GED, each sequence of tokens is prepended with a token [CLS] indicating the beginning of the sequence and add a token [SEP] indicating the separation or end of the first sequence using the tokenizer. The tokenizer allocates each token a unique token ID and gives the information of the tokens that as masked. The BERT-for-Patents model generates a features vector of [CLS] token of a dimension 1×1024 for a 64-token long input sequence. This vector representation of the [CLS] token can be used for various downstream tasks. In our GED model, we perform coarse-grained and fine-grained classification, where we add one and two deep-neural network layers, respectively, to predict the class of error each sentence has. Detailed architectures of our GED models are discussed in chapter 4.

3.4 Computational Linguistics

In this section, we will go through the computational linguistics concepts essential for understanding this thesis's methodology. These techniques assist Specificity (QSM) and Verbosity Model (QVM) in extracting non-lexical information.

3.4.1 Sentence Polarity

Sentence polarity is a form of sentiment analysis, which is a very important topic in the research area of natural language processing. The sentence polarity involves identifying entities conveying positive, negative, and neutral sentiments in a sentence. Sentence polarity is usually performed using supervised methods. However, we have to look for an unsupervised approach when not available with ground truth labels. Various unsupervised approaches are available to find the overall sentiment (polarity) of a sentence based on the polarity of individual words present in the sentence. Some of these methods are - VADER (Valence Aware Dictionary and sEntiment Reasoner) [HG14], SentiStrength [The+10], The General Inquirer [SDS66] and MPQA Subjectivity Lexicon [WWH05]. Although each of these methods computes the polarity of words differently, we will focus on The General Inquirer and MPQA Subjectivity Lexicon in this thesis.

The General Inquirer [SDS66] was introduced to process text to find the use of terms from different categories. It provided the framework for dictionary methods in unsupervised sentiment analysis, which count how many times a list of terms appears in a document. The General Inquirer's original edition included terminology for numerous word classes, such as themes, parts of speech, emotional phrases, and terms for evaluative language. The General Inquirer's initial dictionaries were combined with later dictionaries, and an updated edition was published in the 1990s[2].

A phrase-level sentiment analysis model is used to create the MPQA Subjectivity Lexicon, identifying whether an expression is polar or neutral. Later it separates the polar expressions' positive and negative aspects. The model uses machine learning and several features in a two-step approach. Each sentence that contains an expression is categorized as neutral or polar

[2]Unsupervised Sentiment Analysis, David Garcia, 2021

in the first stage. The second phase emphasizes the contextual polarity of all the terms designated in step one as polar (positive, negative, or neutral) [WWH05].

3.4.2 WordNET

WordNet is a lexical database of semantic relationships between words. WordNET connects nouns, verbs, adjectives, and adverbs through synonyms and other semantic relationships reflecting lexicalized concepts. In WordNET, a language's vocabulary is described as a collection of pairs (f, s) where f denotes an element's form, which is a string over a finite alphabet represented by ASCII letters, and s denotes one of its senses (synonyms) from a given set of meanings. More than $166,000$ (f, s) pairs in WordNET contain $118,000$ different word forms and 90,000 different word meanings [Mil95].

In our thesis, WordNET determines how far a word is from its root word. We can use it to determine whether a word is general or specific. It is based on the notion that the farther a word is from its root word, the more specific it becomes. We first determine the word's POS tag to compute the distance. However, the lemma of two words may be the same but belong to a distinct part of speech in some cases. For instance, although the words *authored* and *author* share the same lemma, but their POS differs. When calculating the distance, we must ensure that the POS tags of the word and the root word match. To calculate the distance, we extract the set of synonyms for the word. We choose the path with the lowest distance from the hypernym paths (example in Figure 3.3) in the synonym set. The root word is closest to the path's lowest point, and the length indicates how far away it is.

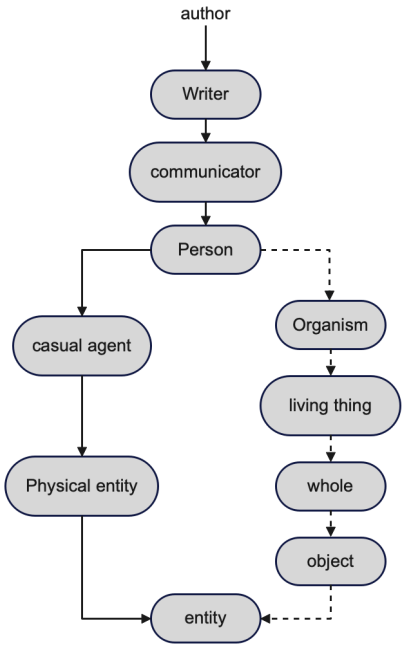

Figure 3.3: Hypernyms paths for the word *author* (noun) from WordNET

3.4.3 Context-free Grammar

Context-free Grammar (CFG) can be defined as the set of rules (productions) for forming well-structured sentences. The grammar contains the syntactical information between the words and phrases of a sentence. These rules have a syntactic category on the left-hand side and alternative component parts on the right-hand side. A series of words that can be derived by systematically applying the rules is called a *sentence*. It begins with a rule that has S on its left-hand side. When a sentence is parsed, it results in a series of rules in which the constituent (syntactic category) is replaced by the right-hand side of a rule with that constituent on its left-hand side [All95]. E.g., for a sentence - "a radiation detection method", the parse tree and the production rules would look like Figure 3.4.

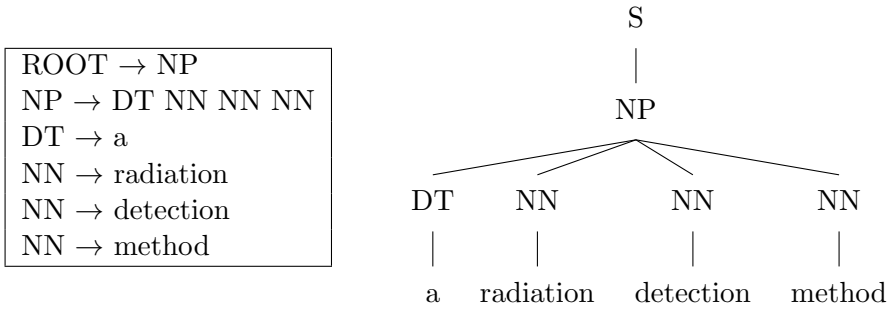

Figure 3.4: Grammar Productions (left) and Parse tree (right) for the given sentence

3.4.4 Constituency Parsing

A constituency tree is a parse tree that entails the syntactical structure of a sentence depending upon formal grammar. It includes the relationships between words or phrases. In computational linguistics, the task of creating a parse tree from a sentence is known as parsing. The constituency parse tree is based on the formalism of context-free grammars (CFG), i.e., the sentence is divided into constituents or categories possessing words and phrases [Eli20]. E.g., *"wearable device with touchscreen"* and *"interaction of two proteins"* belong to the constituent noun phrases (NP). In a constituency parse tree, the words available in a sentence are always included. They are called **Terminals** whereas the constituents are known as **Non-Terminals**. For instance, the constituency parse tree for the sentence - **"The sensor controls electrical energy."** shown in the Figure 3.5. The words in the sentence are the terminals, while the other nodes are non-terminals. The abbreviations are used in the tree shown in Figure **??**:

S	Sentence	NN	Common Noun	DT	determiner	JJ	Adjective
NP	noun phrase	VBZ	Verb	VP	Verb Phrase		

Table 3.1: Abreviations used in the parse tree

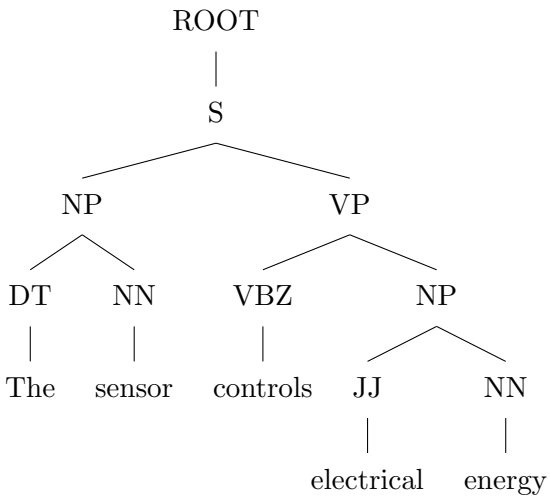

Figure 3.5: Constituency Tree for the given sentence.

3.4.5 Coreference Resolution

Coreference resolution is the task of identifying the mentions of an entity in a text. These mentions can be present in the same sentence, document, and across the documents. Depending upon the task, these mentions are identified and grouped together based on their properties and later replaced with the entity it is talking about. Coreference resolution can be used to solve multiple natural language understanding tasks such as text understanding, information extraction, machine translation, and document summarization. However, the phrase *coreference* is very generalized. There could be many types of coreference resolution. For instance, anaphora resolution, where an entity does not have a direct mention in a sentence or in a document, but the mention represents the entity indirectly using a pronoun or a noun phrase. E.g., in the text shown below 3.6, the word *it* is an indirect mention of *a chair*. In this text *it* is used multiple times with the sentence and also across other sentences, all representing the same entity except in the third sentence. Similarly, in the third sentence, *it* represents the *the height adjustment*.

A chair that estimates the height of the person sitting on it by measuring the angle at which the person's legs are inclined. Then it slowly adjusts its height to a level most comfortable for the user. The height adjustment is so slow that it is cannot be noticed by the person.

Figure 3.6: Mentions of identified entities in the example text

In our thesis, we do not perform coreference resolution, but we use the mentions of entities in or outside the sentence in our query verbosity model 6 to help the model understand the continuity of a sentence. For the continuity features for QVM, we record mentions within and outside the sentence calling them *inter-links* and *intra-links* respectively. In the above

example, *a chair* has one intra-sentence link and two inter-sentence links. Similarly, *the height adjustment* has one intra-link and no inter-links, and *the person* has no intra-link but has an inter-link. Therefore, we have 2 intra-links and 3 inter-links in our text.

3.5 Machine Learning Algorithms

This section will briefly discuss the machine learning algorithms we used to train various models in our thesis.

3.5.1 Linear Regression with Ordinal Least Squares

Ordinary Least squares (OLS) regression is a linear modeling technique using which an individual response variable recorded on an interval scale may be modeled. OLS can be used with single or multiple variables, including categorical values, if they are encoded correctly [Hut11].

The relationship between a dependent variable Y and the independent variable X can be written as the linear equation:

$$Y = \alpha + \beta x \tag{3.1}$$

In the equation, α is the intercept, i.e., the value of Y when X is zero, and β is the slope of the line (or regression coefficient). The regression coefficient β is the change in Y associated with the change in X. The comparison between the predicted values of Y and the true values shows how well the data fits into the model. The difference between the predicted values and the true values (called residuals) gives an indication of how well the model predicts each data point.

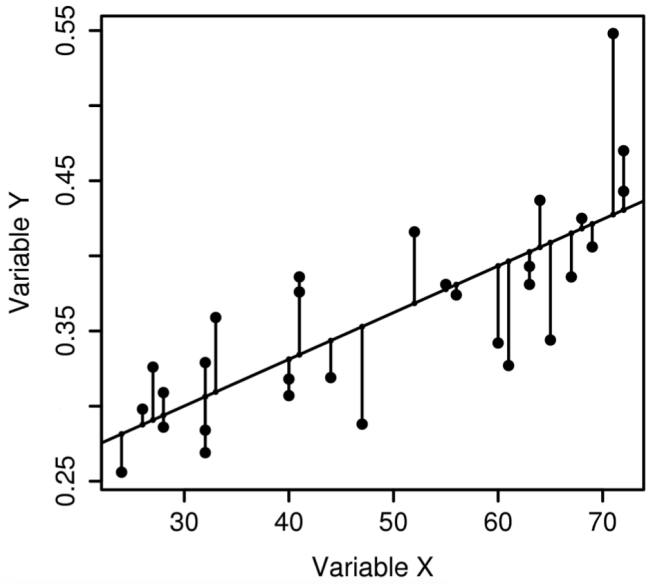

Figure 3.7: OLS Regression Residuals

The sum of squares of the residuals is known as the residual sum of squares (RSS) which provides the measure for model fit for the OLS regression model. For a simple OLS regression model, the effect of an independent variable on Y can be calculated by simply comparing the predictions using equation 3.1 and the null model ($Y = \alpha$). An F-test derived from the following equation can then be used to determine whether the difference in a deviation between the nested models is significant [Hut11].

$$F_{(df_p - df_{p+q}, df_{p+q})} = \frac{RSS_p - RSS_{p+q}}{(df_p - df_{p+q})(RSS_{p+q}/df_{p+q})}$$

where p represents the null model and $p + q$ represents equation 3.1, df are the degree of freedom with the designated model. This equation shows that the F-statistic takes into account the number of parameters while relying on the difference between the deviation of the two models as a percentage of the deviance of the whole model [Hut11]. The R-square statistic, which is frequently used as an example, measures how much variation in the response variable is "explained" by the model [Hut11]. The definition of R-square, often called the coefficient of multiple determination, is:

$$R^2 = \frac{\text{RSS after Regression}}{\text{Total RSS}}$$

3.5.2 Logistic Regression with Liblinear Solver

The probability for classification problems with binary outcomes is modeled using logistic regression. It is an extension of the linear regression model for classification tasks. The logistic regression model uses the logistic function to squish the output of a linear equation between 0 and 1 rather than fitting a straight line, or hyperplane [Mol20]. This logistic function can be defined as:

$$f(x) = \frac{1}{1 + exp(-x)} \tag{3.2}$$

In linear regression, we modeled the relationship between the dependent and independent variables in equation 3.1. Since we prefer the probabilities for classification, then using equations 3.1 and 3.2, we can compute the probability of the outcome:

$$P(y^{(i)} = 1) = \frac{1}{1 + exp(-(\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}))}$$

The binary class logistic regression using Liblinear [Fan+08] optimizes the following cost function as an optimization problem:

$$\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi(w; x_i, y_i)$$

where $C > 0$ is penalty parameter and ξ is the loss function [Fan+08] that is given below:

$$\xi(w; x_i, y_i) = log(1 + e^{-y_i w^T x_i})$$

3.5.3 Support Vector Machine

Support Vector Machines (SVM) is a supervised learning model that could be useful for data classification and regression analyses. SVM aims to find a hyperplane in an N-dimensional space that distinctly classifies the data points. SVM is similar to logistic regression because it is also driven by the linear function. However, unlike logistic regression, it does not provide the class probabilities but only the class identity. Another way SVM is different from logistic regression is that it uses *kernel* trick. The kernel trick is based on the fact that many machine learning models can be written exclusively in terms of the dot product between the examples [GBC16]. The objective of SVM is to pick the plane that maximizes the margin between the data points as shown in Figure[3]. Therefore, the plane that classifies the data with maximum margin is called **Hyperplane**, and the data points falling on either side of the hyperplane are attributed to the participating classes.

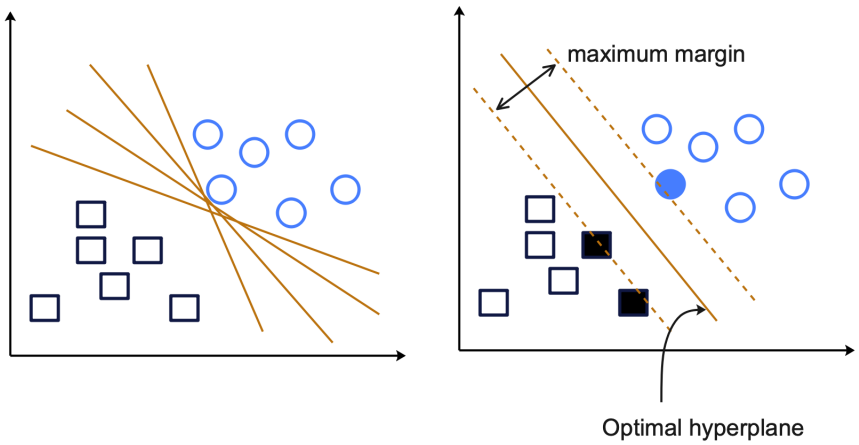

Figure 3.8: SVM Selecting Optimal hyperplane

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane with the maximum margin, i.e., the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified more confidently. Data points closer to the hyperplane help maximize the classifier's margin and influence the hyperplane's orientation and position. These data points are known as Support Vectors. The loss function that helps maximize the loss is Hinge loss. In our thesis, we use SVM for length range prediction of the patent abstracts. We use SVM with radial and linear kernels.

3.5.4 Random Forest

Random Forest is a supervised machine learning technique that can be used for classification regression. The algorithm is based on ensemble learning, a technique of combining multiple classifiers to solve a complex problem. The algorithm involves multiple decision trees.

[3]Figure from SVM:Feature Selection and Kernels

Random forest uses a technique to randomly sample data hence each decision tree is trained independently using sampled data. This technique of selecting a random sample of the dataset is known as **Bagging** or **Bootstrap Aggregation**. The final prediction of the classifier is based on the majority voting by all decision trees. The advantage of the Random forest algorithm over decision trees is that decision trees may suffer overfitting problems if they are allowed to grow without control, but in a random forest classifier, decision trees are trained with a subset of the data sets, and the output is predicted by majority voting hence the problem of overfitting is unlikely to occur.

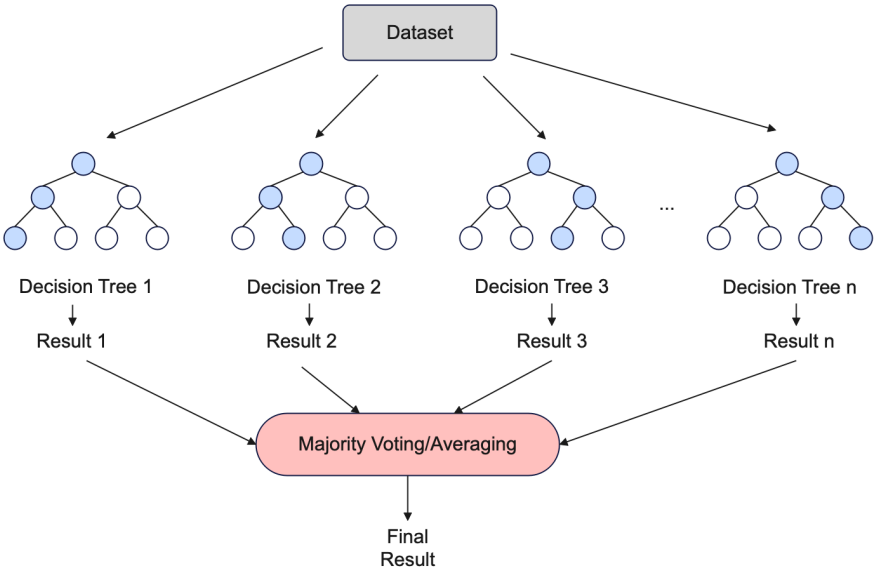

Figure 3.9: Random Forest

3.5.5 XGBoost

Extreme Gradient Boosting, often known as XGBoost, is an ensemble machine learning algorithm that uses decision trees and the gradient boosting framework. In this approach, where weights are crucial, decision trees are generated sequentially. These weights are applied to all the independent variables in the decision trees, predicting the outcomes. Variables that the tree incorrectly predicted are given more weight before being placed into the second decision tree. These distinct classifiers/predictors are combined to produce a robust and accurate model. Regression, classification, ranking, and user-defined prediction problems can all be handled using XGBoost. Although we cannot train several trees simultaneously with XGboost, it can simultaneously create the various tree nodes. Data must be sorted in the proper sequence for that. It stores the data in blocks to lower the cost of sorting. Each column was sorted by the matching feature value, and the data was stored in a compressed column format. By balancing any parallelization overheads in computation, this choice enhances algorithmic performance [CG16].

3.6 Evaluation Matrices

We trained different models in this thesis, and to better understand their performances, we used some evaluation matrices depending on the model type.

3.6.1 Metrics for Classfication Models

3.6.1.1 Confusion Matrix

When describing how a classification model (or "classifier") performed on a set of test data for which the true values were known, a confusion matrix is a table that is frequently used. This Matrix is shown in Figure 3.10. As the table indicates, True positive (TP) are the samples whose true values were 1, and the classifier also predicted 1. On the contrary, False negatives (FN) are instances when the true value was 1, but the classifier predicted it as 0. Similarly, false positives (FP) are the instances when the true value is 0. However, the classifier predicted as 1, True negative were the instances when the true values were 0, and the classifier also predicted 0.

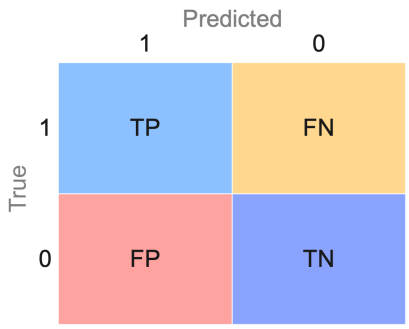

Figure 3.10: Confusion Matrix

3.6.1.2 Accuracy

The accuracy of a machine learning model indicates the proportion of times it will predict a result accurately out of all the predictions it has made. Model accuracy is the proportion of true positives and negatives to all positive and negative data.

$$\boxed{\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}}$$

3.6.1.3 Precision

The model precision score is a measure of the proportion of correctly predicted labels that are correct. Precision is often referred to as the positive predictive value. It shows the proportion

of true positives to the total of both true and false positives.

$$\boxed{\text{Precision}(P) = \frac{\text{TP}}{\text{TP} + \text{FP}}}$$

3.6.1.4 Recall

Recall score measures how well a model can predict positive results from actual positive results. It measures how well our machine learning model can distinguish between a dataset's true positives and false positives. The greater the recall score, the better the machine learning model recognizes positive and negative examples. It shows how many true positives there are compared to the total of true positives and false negatives.

$$\boxed{\text{Recall}(R) = \frac{\text{TP}}{\text{TP} + \text{FN}}}$$

3.6.1.5 F1-Score

A machine learning model's performance is measured using the F-score metric, which weighs Precision and Recall equally. It can be formulated mathematically as a harmonic mean of precision and recall score.

$$\boxed{\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}}$$

3.6.1.6 Micro vs. Macro Averaging

In a multiclass or multilabel classification problem, the precision and recall scores are computed differently because there are multiple classes to be predicted by the classifier. Depending upon the samples available for each class, simply averaging the scores is not advised. Therefore, in such scenarios, there are two possible ways of averaging precision and recall scores for multiclass problems, i.e., macro and micro-averaging. In macro-averaging, the metric is computed independently for each class, and then those scores are averaged. E.g. for classification with N classes, micro-averaged precision can be calculated as:

$$\boxed{P_{macro} = \frac{1}{N} \sum_{i=1}^{N} P_i}$$

In micro-averaging, the contribution of each class is aggregated to calculate the average. E.g., micro-averaged precision for N classes can be computed as:

$$\boxed{P_{micro} = \frac{\sum_{i=1}^{N} \text{TP}_i}{\sum_{i=1}^{N} \text{TP}_i + \sum_{i=1}^{N} \text{FP}_i}}$$

Similarly, recall scores can be calculated.

3.6.1.7 Exact Match Ratio

In a multi-label classification scenario, an exact match score is the ratio of instances when the classifier predicted all labels exactly like the true labels. This measure does not include partially correct examples [Sor].

$$\boxed{\text{Exact Match Ratio} = \frac{1}{n}\sum_{i=1}^{n} I(y_i = \hat{y}_i)}$$

Here I is the indicator function.

3.6.1.8 Hamming Loss

Hamming loss reports the proportion of times the classifier predicted an incorrect label, for example. The hamming loss considers the labels incorrectly predicted or not predicted at all [Sor][Ped+11].

$$\boxed{L_{Hamming}(y, \hat{y}) = \frac{1}{n_{\text{labels}}} \sum_{j=0}^{n_{\text{labels}}-1} 1(\hat{y}_j \neq y_j)}$$

3.6.1.9 Jaccard Score

The Jaccard similarity (or score) calculates the similarity between two data collections to identify common and unique individuals. The Jaccard similarity is determined by dividing the total number of observations in both sets by the total number of observations in either set [Ped+11].

$$\boxed{J(y_i, \hat{y}_i) = \frac{|y_i \cap \hat{y}_i|}{|y_i \cup \hat{y}_i|}.}$$

3.6.2 Metrics for Regression Models

3.6.2.1 Pearson Correlation Coefficient

Pearson Correlation summarizes the strength of the linear relationship between two data samples. It returns values between -1 and 1. If it returns 0, that indicates no correlation, for -1 and 1 mean full negative and positive correlation, respectively.

$$\boxed{r_{Pearson} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}}$$

3.6.2.2 Spearman's Correlation

Spearman rank correlation is a non-parametric test used to assess how closely two variables are related. When the variables are measured on a scale that is at least ordinal, the Spearman rank correlation test is a suitable correlation analysis because it carries no assumptions about the distribution of the data.

$$\boxed{\rho = 1 - \frac{6 \sum D^2}{n(n^2 - 1)}}$$

3.6.2.3 Kendall Rank Correlation

Kendall rank is also a non-parametric test that measures the strength of dependence between two variables.

$$\boxed{\tau = \frac{n_c - n_d}{0.5 \times n(n-1)}}$$

4 Grammatical Error Detection

Grammatical error detection (GED) is a natural language processing task that can be used to identify a text with grammatical errors. This model can be used as a predecessor of Grammatical error correction. However, this thesis restricts itself only to GED.

Although, the patents are filed all across the world, even in countries where English is not a common language. However, it is a fact that not all patents are filed in English; some of them are also written in languages like German, French, Mandarin, etc., but most number patents are either written in English or they are translated into English. Usually, those patents are written by language experts on behalf of the inventor. After that, it passes through multiple iterations of pointing out errors, including grammatical errors. Hence, it is highly unlikely that a patent might contain grammatical errors.

Query 1: *a stationary object*	**Query 2:** *a stationery object*
Top 5 Results	**Top 5 Results**
1. Methods and compositions for non-invasive dose-to-effect administration of drugs with cardiovascular or renal vascular activities	1. Watching bird novelty item
2. Electronic apparatus, information processing device, and information processing method	2. Value dispensing mechanism, such as a postage meter, having automatic display/printing selection
3. Display device with anthracene and triazine derivatives	3. Display stand
4. Touch display apparatus	4. Information organization product and method
5. Electrolyte for stable cycling of high-energy lithium sulfur redox flow batteries	5. Information processing apparatus, method therefor, and computer program

Table 4.1: Results provided by PQAI when using homonyms

But when it comes to searching for prior art, initially, the search is performed by the inventor himself/herself. If it is a natural language query, it is possible that a person whose first language is not English might commit mistakes forming a prior art search query. As a result, the inventor might end up considering wrong and irrelevant prior art. Hence, the developer should emphasize this fact when developing a prior art search engine. A well-written prior art search query should be unambiguously parsable by the search engine to reveal the underlying interrelationships among its constituent concepts. One may ask this question, a search engine works by indexing the entities given in a query and returning the entity pointers. When everything happens on an entity or phrase level, why is there a need for grammatical error

detection? This question makes sense as long as the inventor makes no mistake in writing the phrases. For example, if we compare these two phrase: *a stationary object* and *a stationery object*. Both look like similar phrases, but a search engine would provide two completely different results. We used these two phrases as queries on PQAI prior art search engine, and the top 5 results of both searches can be seen in Table 4.1.

Queries	**No. of matching results**	**Order Changed?**
Error 1: Incorrect Verb Form **Correct:** a radiation detection method and apparatus thereof for estimating radiation power received by the mobile station **Incorrect:** a radiation detection method and apparatus thereof for estimated radiation power receiving by the mobile station	5	Yes
Error 2: Incorrect article Error **Correct:** a cutter head assembly comprising a first longitudinal member **Incorrect:** the cutter head assembly comprising an first longitudinal member	6	Yes
Error 3: Singular/Plural error **Correct:** the invention relates to an electronic component with a rich deposit layer on the part for electric connection **Incorrect:** the invention relates to an electronic components with sn rich deposits layer on the part for electric connections	3	Yes
Error 4: Similar word error **Correct:** a stationary object **Incorrect:** a stationery object	1	Yes
Error 5: Spelling error **Correct:** a power supply for an electrical appliance has a female connector for an electrical plug **Incorrect:** a power supply for an electricle appliance has a female connecter for an electrical plug	1	Yes

Table 4.2: No. of matching results by PQAI when correct and incorrect queries are searched.

Similarly, we performed a prior art search using five different queries having each type of five errors and their equivalent correct query. Table 4.2 showed the number of the same results when both correct and incorrect queries were also performed and if the matching results' priority had changed.

It is visible from Table 4.2 that a small grammatical error in a query can change the results drastically. However, it is a point of discussion if the results provided by the search engine

are relevant to the query or not in both cases. We will discuss that in further chapters. For the time being, our only focus is to point out grammatical errors. It would surely make a difference in the text quality of the query and, ultimately, in the search results.

4.1 Grammatical Error Data Generation

As we discussed in the previous section, a patent is filed after multiple iterations of checks and rectifying errors. So the probability of finding grammatical errors in a patent document is almost zero. Hence, it is evident that we will be unable to find any data source from where we can find a patent dataset with grammatical errors in the text. Therefore, ultimately we would need to make such a dataset by ourselves.

Sharma et.al [SLW19] introduced a large-scale summarization dataset consisting of 1.3 million U.S. Patent documents collected from Google public datasets. It contains patents from nine technological areas. We collect a subset of this dataset, having 12375 documents in the train, 697 in the test set, and 688 in the validation set. We use abstract texts of this dataset to introduce grammatical errors.

4.1.1 Types of Grammatical Errors

According to a document named Common Grammatical Errors[1] shared by the University of Technology, Sydney, there are 11 common mistakes made by people while writing an essay. However, some of them are related to the placement of punctuation. For this task, we will not introduce errors caused by punctuation. We have chosen five major grammar errors from these 11 because we think they are likely to be made by users when constructing a search query and will negatively impact search results. These error types are the following:

4.1.1.1 Incorrect verb form

There are two possibilities if the wrong verb is used in a sentence.

1. Incorrect subject-verb agreement: Wrong relationship between a subject and its verb. For example,

Incorrect: *She prefer coffee over tea.*
Revised: *She prefers coffee over tea.*

2. Wrong tense or verb form verb form that indicates whether you are referring to present, past or future. E.g.,

Incorrect: *Because I worked the whole day, I sleep early.*
Revised: *Because I worked the whole day, I slept early.*

[1]Common Grammatical Errors by University of Technology, Sydney

4.1.1.2 Singular/Plural error

Agreement between the noun forms. E.g.,

Incorrect: *There are so many grammatical error in that document.*
Revised: *There are so many grammatical errors in that document.*

4.1.1.3 Incorrect article error

The incorrect use or omission of *a*, *an* and *the*.

Incorrect: *I eat a apple every morning.*
Revised: *I eat an apple every morning.*

4.1.1.4 Incorrect word use (Homonyms) error

A word that is pronounced similarly to the other word but has a completely different meaning is used in a sentence. E.g.,

Incorrect: *Every group should contain equal number of mails and females.*
Revised: *Every group should contain equal number of males and females.*

4.1.1.5 Spelling errors

A word is misspelled. E.g.,

Incorrect: *Mount Everest is the highest mountane in the world.*
Revised: *Mount Everest is the highest mountain in the world.*

4.1.2 Introducing Grammatical Errors in the Dataset

We have set a set of abstract documents $D = \{d_1, d_2, d_3...d_i\}$. We introduce each error in 30% documents of D.

4.1.2.1 Incorrect Verb form Error

To introduce this type of error in a set of abstract documents D, we identify all the verbs $v(d_i)$ present in an abstract text d_i. We generate a random number r between zero and j where $j = Count(v(d_i))$. We select verbs equal to r and replace all verbs randomly selected with one of their lexemes. We have,

$$v(d_i) = \{v_1, v_2, v_3...v_j\}$$

Calculating random number r,

$$r = random(0, j)$$

Randomly select r verbs from $v(d_i)$,

$$\bar{v}(d_i) = \{\bar{v_1}, \bar{v_2}, \bar{v_3}, ...\bar{v_r}\}$$

and lexeme of $\bar{v_r}$,

$$l_v^r = \{lv_1, lv_2, lv_3, ...lv_n\}$$

then,

$$\bar{v_e}(d_i) = \{rand(l_v^1), rand(l_v^1), rand(l_v^2), ...rand(l_v^r)\}$$

where
$\bar{v_e}(d_i)$= error introduced in the verb of d_i

4.1.2.2 Singular/Plural Error

Similarly, we introduce singular/plural errors in the abstract document set D. We identify all the nouns and pronouns from the document d_i. We generate a random number r and select r nouns and pronouns based on the count of the set of nouns and pronouns. For every noun or pronoun in the selected set, we replace it with its subsequent plural or vice versa.
We have,

$$np(d_i) = \{np_1, np_2, np_3, ...np_j\}$$

After generating a random number r, and selecting r nouns/pronouns, we have,

$$\bar{np}(d_i) = \{\bar{np}_1, \bar{np}_2, \bar{np}_3, ...\bar{np}_r\}$$

using function

$$f(x) = \begin{cases} Plural(x), & \text{if } x \text{ is singular} \\ Singular(x), & \text{otherwise} \end{cases}$$

We generate errors in the $\bar{np}(d_i)$. Therefore, the noun-pronoun error set,

$$\bar{np}_e(d_i) = \{f(\bar{np}_1), f(\bar{np}_2), f(\bar{np}_3), ...f(\bar{np}_r)\}$$

4.1.2.3 Incorrect article Error

Incorrect article error is applied to r determinant terms of a document $d_i \in D$. Number r is a random number between zero and the document's total number of determinant terms. This set of r determinant terms can be written as,

$$\bar{dt(d_i)} = \{\bar{dt}_1, \bar{dt}_2, \bar{dt}_3, ...\bar{dt}_r\}$$

We know that there are 3 three determinant terms i.e., *a*,*an* and *the*. Hence, set A has all these terms.

$$A = \{a, an, the\}$$

Therefore, using function

$$f(x) = \begin{cases} x \leftarrow rand(A), & \text{if } rand(A) \neq x \\ f(x), & \text{otherwise} \end{cases}$$

our new set of determinants for document $d_i \in D$.

$$d\bar{t}(d_i) = \{f(d\bar{t}_1), f(d\bar{t}_2), f(d\bar{t}_3), ...f(d\bar{t}_r)\}$$

4.1.2.4 Incorrect word use (Homonyms) error

Pitt et al. [PE02] provide a brief list of homonyms used in the English language. This list contains 498 unique words that have one or more homonyms and 215 unique combinations of homonyms. To introduce this type of error in the text $d_i \in D$, we replace all the common words in the text and in our set of homonyms and replace them with another word from that homonym combination. We have a set of homonym combinations and a flat set,

$$H = \{(h_1, h_2, h_3)_1, (h_1, h_2, h_3)_2, ...(h_1, h_2, h_3)_j\}$$

$$H_{flat} = \{h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, ...h_{j1}, h_{j2}, h_{j3}\}$$

and the document $d_i \in D$,

$$d_i = \{w_1, w_2, w_3...w_n\}$$

Then for every word $w_n \in H_{flat}$, we locate the combination j of H_{ji} in H and replace it with another word in the combination.

4.1.2.5 Spelling Errors

Flor et. al [FFR19] introduce a dataset called TOEFL-Spell that contains Spelling Annotations for English language learner essays written for TOEFL exams. This dataset contains more than 6000 spelling errors from the essays of non-native English speakers taking TOEFL exams. We use this dataset to introduce spelling errors in our dataset. For a document $d_i \in D$, we identify words present in the spelling error dataset se, which contain words and their misspelled version. Depending upon a random number r generated between zero and the number of words available in both the spelling error dataset and the document, we pick r words from the document and replace them with their misspelled form.
We have a spelling error set,

$$se = \{(w, m)_1, (w, m)_2, (w, m)_3, ...(w, m)_n\}$$

$$se_{words} = \{w_1, w_2, w_3, ...w_n\}$$

and set of common words in se_{words} and the document d_i,

$$c(d_i) = d_i \cap se_{words}$$

a portable device foir secured loads , includ the plurality pf articles, In motor vehicals, which device comprising a platform meant having hook - llike mesh engaging elements providing along oin edge thereof, and net means sicure along a oopsite edge of saied platform means so that meshes of say net qre engaging bzy saying hook - like elements when sais net is pull taut over articles places on saied platform means.

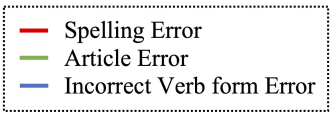

Figure 4.1: An abstract document with multiple errors

Based on a random number $r = random(0, Count(c(d_i)))$, we select r words from $c(d_i)$. Hence subset $c(\bar{d}_i) \subset c(d_i)$. After that, we replace all the words from $c(\bar{d}_i)$ to their misspelled form.

$$c_n(\bar{d}_i) \leftarrow m(se_n)$$

where $m(se_n)$ is the misspelled form of word at index n in se set.

4.1.3 Labels

We encode all five errors mentioned in the previous section, each with a value of 0 or 1. Value 0 shows that a particular error is absent, whereas 1 indicates that the error is present in the document. The sixth label symbolizes whether the document is grammatically correct or not. Hence, for each document $d_i \in D$, we have a set of six labels of dimension 1×6.

4.1.4 Dataset with Grammatical Errors

The finalized dataset for grammatical error detection has 12375 abstract documents in the train, 697 in the test, and 688 validation set. Amongst all these three datasets, we have introduced errors in the 30% document of each set. Hence, there are 3712, 209, and 206 documents with errors in train, test, and validation datasets, respectively. Figure 4.1 shows a document from the train set having multiple errors.

4.2 Grammatical Error Detection Model

A grammatical error detection model helps us identify errors in a text document. In this thesis, we use abstract texts from BIGPATENT dataset [SLW19] to perform error detection. However, this model is about detecting the errors, not correcting them. Thus it makes a classification task. We use two classification models to detect documents with errors in the patent text. First, called *Coarse-grained Classification*, which predicts whether a document

is grammatically correct or not. The Second classification model, called *Fine-grained classification*, predicts if a particular type of error is present in the document or not. These classification approaches are discussed briefly in the background chapter 3.

4.2.1 Data

Using methods discussed in 4.1.2, we introduced grammatical errors in the abstract part of BIGPATENT [SLW19] dataset. As the dataset is split between the form of training, validation, and test sets, every set contains errors in 30% of the documents. The distribution of documents with errors and without errors in all three sets is shown in fig 4.2.

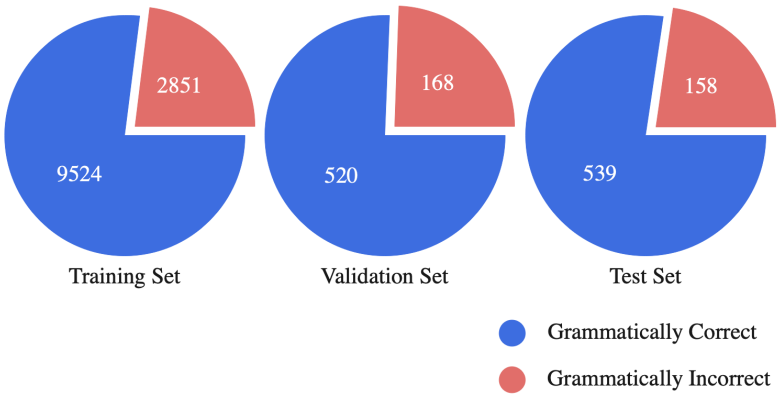

Figure 4.2: Distribution of error documents in the dataset for GED

4.2.2 Coarse-grained Classification Model

The coarse-grained classification model is a binary classification task that predicts if a text is grammatically correct or not. To begin with, we tokenize the abstract text from the training set. Since we are using a pretrained model for this task, we need to ensure that the input data is in the same format as the format used for training the pretrained model. We utilize the tokenizer of a pretrained BERT model called BERT-for-Patents. The tokenizer splits the sentences into sequences and maps a token ID with each token. For each text, We add special tokens like [CLS] and [SEP] in the texts. The token [CLS] denotes the beginning of the text, whereas [SEP] is added to help the model understand which token belongs to which sentence. We apply padding and truncating on all the texts to the maximum length of 64. That means if a text has a length of 64, it will add the additional token to the text sequence. If the sequence length exceeds 64, it will remove the extra tokens. After tokenization of the text, we have *input_ids*, containing token ids of all text sequences, and attention_masks, having information of padding applied to the sequence. We fine-tune the BERT-for-Patents model in batches.

We select a set of three different batch sizes, i.e. $[16, 32, 64]$. Since there are two classes to be classified in this task, we use the binary cross entropy loss function. We use adam optimizer

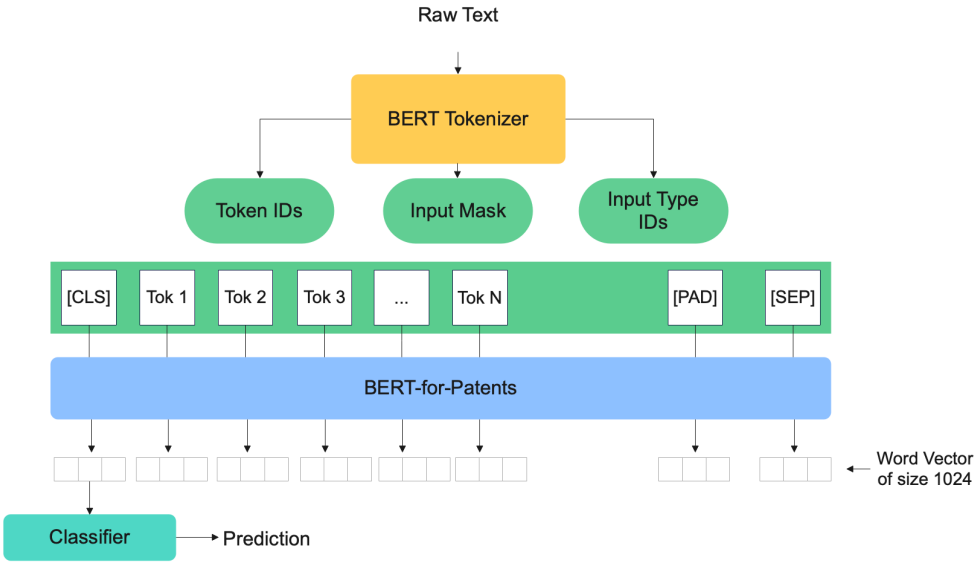

Figure 4.3: Architecture of GED-CG classification model

to calculate this model's gradients with four different learning rates, i.e., $[2e-5, 1e-5, 3e-4, 5e-4]$. We train the model for a set of three epoch values, i.e., $[2, 3, 4]$. Loss and accuracy are recorded on the validation set after every epoch. The results section discusses this model's performance and parameter tuning briefly.

4.2.3 Fine-grained Classification Model

The fine-grained classification model for grammatical error detection is a multilabel classification model that predicts if a particular type of error exists in the corpus or not. The dataset for grammatical error detection has five types of errors in a document discussed in the previous sections. The labels for each patent abstract are a 1×5 matrix indicating the presence or absence of an error by 0 or 1, respectively. One may ask why there is a need to point out a mistake if we have a well-performing model that classifies correct and incorrect documents. Although for this thesis, our task is only to differentiate the two types of texts, it would be beneficial for further expansion of the PQAI prior art search engine. It would let the user know which particular term in a query he/she is making a mistake. Another advantage of training this model would be to help users correct their errors by suggesting options for selecting the right term. To train this classification model, we perform feature extraction steps the same as in the coarse-grained classification model 4.2.2. We use the tokenizer of the pretrained model BERT-for-Patents, which splits the sentences into tokens and allocates a token ID to each token. The maximum length of a document is kept at 64. Hence if the text has more than the full length, we omit the extra words, and if there are fewer words, additional padding is added to that sentence. In the end, have three sets, one containing attention ids containing the information of the tokens where padding is applied, the second is a matrix of 12375×64 containing token IDs of the sentences, and a label matrix of 12375×5. The architecture of this model is a little different than the coarse-grained classification model.

The first is a BERT layer that expects embedding information from a pretrained model.

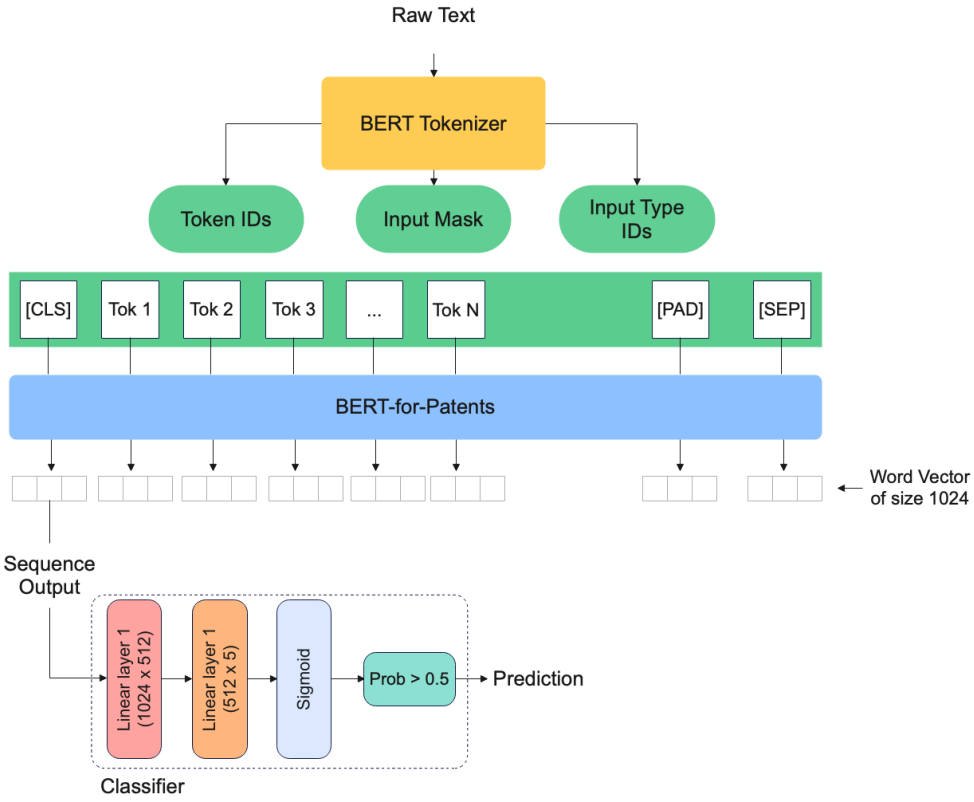

Figure 4.4: Architecture of GED-FG classification model

We use the same pretrained model called BERT-for-Patents. The Bert layer loads the weights from the pretrained model, and the model transformer outputs raw hidden states without any specific head on top. We use the last hidden state's output as an input for our next layer. We add two linear layers with 1024 and 512 neurons, respectively. The sigmoid activation function is applied to the second linear layer's output to compute each label's probability score. We use binary cross-entropy to calculate the loss between the true and predicted labels. We use the Adam optimizer of compute gradients with a set of learning rates $[2e-5, 1e-5, 5e-4]$. We train the model in batches of sizes $[32, 64, 128]$ for $[1, 2, 3, 4]$ epochs.

4.2.4 Baseline

Kim et al. [Kim14] introduced a sentence classification model using a Convolution Neural Network (CNN). This model uses $k-$dimentional word vectors corresponding to each word in a sentence. A filter is applied to a window of words for producing new features as a convolution operation. Then they use the max-pooling function over the feature map and take the maximum value as the feature corresponding to this particular filter. The model uses multiple such features, and their output is passed to a fully-connected linear layer, and at the end, there is a softmax layer. They employ drop out at the first linear layer with the

constraint on l_2 norms of weight vectors [Kim14]. The architecture of this model is shown in figure[2] 4.5. This model uses 100-dimensional patent domain-specific word vectors described in Risch et al. [RK19].

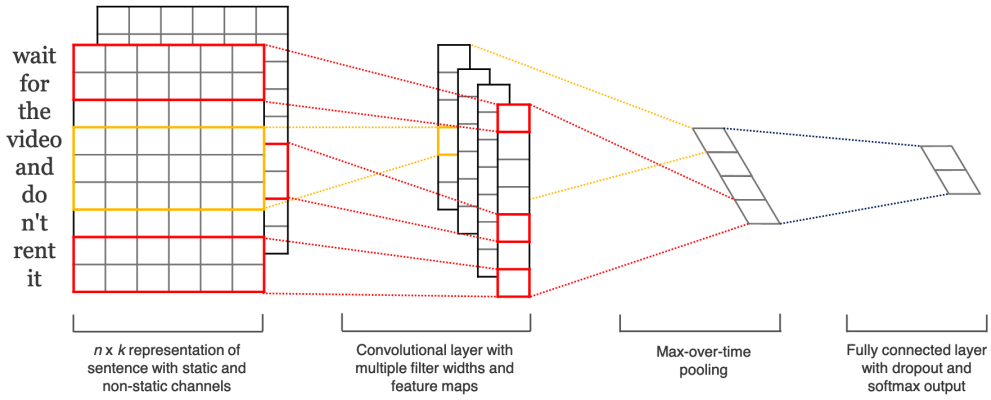

Figure 4.5: Baseline: Model architecture with two channels for an example sentence.

[2]Figure 4.5 by Kim et al. [Kim14]

5 Query Specificity

Query Specificity Model (QSM) is a supervised classifier that differentiates general and specific sentences in a text. In academic writing, it has been observed that the introduction and conclusion sections of a paper contain more general sentences than specific ones, whereas the abstract is more informative and very specific with details [Lou13]. If we look at a text in general, it is a fact that not all sentences in a text convey the information similarly. Sentences in the beginning usually give an overview of the text, and more detailed information comes in further sentences. Similarly, a natural language prior art search query might contain an overview initially, but it has to be more specific for a prior art search engine to provide relevant results. Hence, a query, or any text in general, is a mix of general and specific sentences. But we hypothesize that the prior art search query should contain more specific sentences than general ones.

In this part of our thesis, we focus on detecting general and specific sentences in a query. The task of annotating general vs. specific sentences in a text from the patent domain is not available. However, the same has been done on news corpus in Louis et al. [LN11]. Their approach first exploits the indirect annotations from certain types of discourse relations and trains their model, considering one sentence as general and the other(s) as specific. We adapt their approach for annotating prior art queries to identify the same in prior art queries. We also look for other patterns that their approach misses point out.

5.1 Data

We use the Penn Discourse Treebank (PDTB) dataset [Pra+08], which has over 1 million words from Wall Street Journal (WSJ) articles. We use implicit relations from this dataset. These sentences are not linked by an explicit discourse connective such as *because* or *but* [LN11]. We consider only Instantiation and Specification for this task. The idea is to use the first sentence as general and the second as specific. Table 5.1 shows an example of Instantiation and Specification sentences. In the PDTB dataset, we select these sentences and their full raw texts. Instantiation and Specification relations are kept separately for training two classifiers. But we would also combine them and train the third classifier. We split each of these relations into two parts: the first represents a general sentence, and the other is specific. Some of these relations have more than two sentences, but we will simply remove such examples. We have 1271 relations in the Instantiation dataset and 2139 in the Specification. We split each of these relations into two parts: the first represents a general sentence, and the other is specific. Some of these relations have more than two sentences, but we will simply remove such examples. We have 1271 relations in the Instantiation dataset and 2139 in the Specification.

Instantiation: *Image has, of course, a great deal to do with what sells and what doesn't, and it can't be forced. Wine merchants can't keep Roederer Cristal in stock, but they have to push Salon le Mesnil, even lowering the price from 115 to 90*
Specification: *An ambitious expansion has left Magna with excess capacity and a heavy debt load as the automotive industry enters a downturn. The company has reported declines in operating profit in each of the past three years, despite steady sales growth*

Table 5.1: An example of Instantiation and Specification statements

5.2 Feature Extraction

We use these relations to study the properties of general and specific sentences to determine whether a model can differentiate between the two. Louis et al. [LN11] experimented with 10 of both types of relations and came up with several features that could help differentiate the two types of sentences. Some of the features require non-lexical information, whereas some of them need lexical information from the sentences. We extract the following features from general and specific sentences in both Instantiation and Specification relations for each relation.

5.2.1 Sentence Length

Generally, general sentences do not require too many details to explain or specify things to add more emphasis; however, specific sentences do. Therefore a general sentence is expected to contain fewer words than a specific sentence. Therefore, we calculate the number of words in each sentence.

5.2.2 Polarity Features

Louis et al. [LN11] found out that sentences with strong opinions are general. Therefore, we record the polarity features using two lexicons, i.e., The General Inquirer [SDS66], and The MPQA subjectivity Lexicon [WWH05]. Using both of these lexicons, we record the number of positive and negative words in each sentence. If the count of positive words is P_{s_i} and the count of negative words is N_{s_i} in a sentence s_i, the polarity Score PS is calculated using the following formula for a sentence:

$$Polarity = \frac{P_{s_i} - N_{s_i}}{P_{s_i} - N_{s_i}}$$

We also record the normalization score (N) using the number of positives and negatives concerning the sentence length l_{s_i}. This score would help the model understand the overall

sentiment of the sentence.

$$N = \frac{l_{s_i} - Min(P_{s_i}, N_{s_i})}{Max(P_{s_i}, N_{s_i}) - Min(P_{s_i}, N_{s_i})}$$

5.2.3 Specificity Features

It is obvious that specific sentences would contain more specific words than general sentences. To capture the specificity among the sentences, we calculate the distance of a noun and verb to its root word in WordNet [Mil95]. It works based on a hypothesis that the more the distance of a word is from its root, the more specific it is. The minimum, maximum, and average of the distances of words in sentences are used as specificity features. Another measure is calculating the terms' Inverse Document Frequency (IDF). IDF is a measure that indicates how commonly a word is used in a document. It is used to boost the scores of the word that are unique to a document in the hope that they will help characterize the specific sentence by lowering the IDF of frequent words. For this task, instead of using articles from the news article dataset, we use abstract sentences of the BIGPATENT [SLW19] dataset to compute IDF. This can be calculated using the following formula.

$$IDF_w = \log \frac{N}{n}$$

N is the number of abstract texts in the BIGPATENT dataset, and n is the number of texts where the word w appears. We use the sentence's minimum, maximum, and average IDF values as features.

5.2.4 NE+CD Features

Patent articles sometimes might contain named entities or numbers. Therefore, count named entities and proper nouns from a sentence. According to Louis et al. [LN11], usually, general sentences contain plural quantities. Therefore, we use the number of plural nouns as a feature.

5.2.5 Syntactic Features

Louis et al. [LN11] observed that general sentences contain more qualitative words like adjectives and adverbs. Therefore, we include syntactic features like the count of adjectives, adverbs, adjective phrases, and adverbial phrases. We also calculate the number of verb phrases, their average word length, and the number of prepositional phrases.

5.2.6 Language Model Features

We build unigram, bigram, and trigram models utilizing BIGPATENT abstracts to capture random and catchy words and phrases from the general sentence. Using each model, we estimate the log probability and perplexity of the sentences and use them as features. To build the three language model, we extract all possible unigram bigram and trigram of the abstract sentences of the BIGPATENT dataset. Based on the occurrence of n-grams, we compute the

probability of each. Also, all the unigrams, bigrams, and trigrams of the Instantiation and Specification dataset are extracted. Based on the probabilities of n-grams in BIGPATENT sentences, we predict the word sequence from our general and specific sentences. We use the following formulae to predict the probability of a sequence from a general/specific sentence. For the unigram model,

$$P(w_1, w_2, w_3, ...w_n) = \prod_{i=1}^{n} P(w_i)$$

and bigram model,

$$P(w_n|w_1, w_2, ...w_{n-1}) = \prod_{i=1}^{n} P(w_i|w_{i-1})$$

Similarly, trigram model,

$$P(w_n|w_1, w_2, ...w_{n-1}) = \prod_{i=1}^{n} P(w_i|w_{i-2}, w_{i-1})$$

We can calculate perplexity using the following formula (for bigram),

$$PP(W) = P(w_n|w_1, w_2, ...w_{n-1})^{\frac{1}{N}}$$

We use log probability and perplexity of sentences from all n-grams as features.

5.2.7 Word Features

We collect the count of words in a sentence as features. Before estimating the count of words, we remove non-alphabetical characters like numbers and punctuation from the sentence. Here we use CountVectorizer from SKlearn [Ped+11] that converts the collection of sentences to the matrix of token counts.

5.3 Specificity Model

After extracting features from the sentences, for Instantiation, we have 38 non-lexical features and 8165 lexical features (word counts), whereas, for Specification, we have 38 non-lexical features and 10075 linguistic features. The reason behind the different number of non-lexical features could be that there are more words in the Specification dataset than Instantiation. Upon combining all the components for each dataset, we have feature sets of size 2542×8203 and 4278×10113, respectively. For each sentence in both datasets, we label them with 0 or 1 depending upon if the sentence is general or specific. It should be noted that no pairing information of general and specific sentences was preserved. Therefore, it becomes a binary classification task. Unlike Louis et al. [LN11], We build three classifiers instead of 2. The first classifier is trained using data from Instantiation sentence features, while the second classier is trained with Specification data. The third classifier is trained by combining both datasets. In all three cases, the data is split randomly into train and test sets where the test set contains 20% of the data. For the classification, we use the Logistic Regression model from ScikitLearn

[Ped+11] using *liblinear* solver. We evaluate the performance of the classification using 10-fold cross-validation. Louis et al. [LN11] state that the reason behind using the logistic regression model is that the probability measure would be more appropriate to associate with each sentence rather than hardcore classification into two classes.

Some features contribute more than others to the successful classification. In the results section, we perform a detailed feature analysis to see which types of features better distinguish between general and specific sentences. We evaluate the performance of the three models using the matrices like accuracy, precision, recall, and F1-score. Our model predicts new sentences in the form of probabilities. For the time being, we keep the threshold of 0.5 to determine the labels, but later on, we will plot a ROC curve to decide the optimum threshold value. Additionally, it would be essential to know how the model trained with Instantiation and Specification sentences combined would perform because if we look at both datasets, they contain very similar sentences except for some structural differences discussed earlier. We expect this model to perform better because it was trained on more data than the other two. Another reason is that the model will be trained on a variety of data due to the slight variations in the two datasets.

5.4 Annotation of BIGPATENT Sentences

With a model trained using discourse relations from PDTB sentences, we differentiate general and specific sentences in BIGPATENT abstracts using our best-performing model. We conduct a survey asking seven participants with an excellent understanding of the English Language to mark 20 sentences as general or specific. However, we do not disclose the predictions of our model for these sentences to the participants. This survey would help us evaluate our model using human annotations as ground truth.

6 Query Verbosity

The term verbosity can be defined as the property of a text that contains irrelevant details than necessary. As a result, the text seems longer, and the information it wants to convey starts looking fabricated. Therefore, that makes the document of lower quality and unpleasing to readers. On the contrary, a concisely written document contains the right amount of details necessary for the reader to know. Verbosity in a document arises for two reasons: first, when a document contains irrelevant information, and the second, when a document has redundant information [Wil90]. Irrelevant information is the words that do not play an important role in the text. Even after they are removed from the text, they will give the same information to the reader. Table 6.1 [Wil90] shows an example where the first sentence contains many words that look like fillers after reading the second sentence.

Original Text:
For all intents and purposes*, American industrial productivity* ***generally*** *depends on certain factors that are* ***really*** *more psychological* ***in kind*** *than of any* ***given*** *technological aspect.*
Modified Text:
American industrial productivity depends more on psychology than on technology

Table 6.1: Text having irrelevant details, and its possible modified version

In the English language, it has become a practice to combine similar or the same words, e.g., *each and every*, *basic and fundamental*, *and so on and so forth*, etc. These words are very similar to each other, and they do not make more sense in a sentence. Similarly, some words indicate their general categories. For example, table 6.2 shows a sentence where from the original sentence, a reader can easily infer the *Education* is a process and *athletic* is an activity. But still, this pattern can be noticed in many texts.

Original Text:
The educational ***process*** *and athletic* ***activities*** *are the responsibility of* ***county*** *governmental systems.*
Modified Text:
Education and athletics are the responsibility of county governments

Table 6.2: Hidden redundancy in a text

These patterns of verbosity are not only limited to the texts written by a novice but also by some expert writers who write like that. In patent documents also, these patterns are visible.

However, we cannot call it mistakes, but they make a document longer hence introducing verbosity in the document. Hence, from the examples above, we can conclude that the impact of verbosity on a document is that it increases its length.

We expect that the user's query would have these patterns while forming a prior art query. When searching prior art, such a query would impact the search process due to the fact that the model would have to process unnecessary words ending up increasing the complexity and processing time. In chapter 4, we saw that PQAI prior art search engine is very sensitive to the words and their occurrences. One mistake in a search query affects the entire set of results. We collected a few queries that we believe are verbose according to the standard definition of verbosity and their subsequent queries that are not verbose. We tested out these queries on PQAI. In this chapter, we will only discuss verbosity caused by irrelevant details. Hence, we will use verbosity as irrelevant details in all the examples. Table 6.7 shows verbose queries and their subsequent well-written form and the number of matching results in the set of the first ten results provided by PQAI.

Verbose Query	**Well-written Query**	**Matching Results (Out of 10)**
a smartwatch that determines that the wearer is asleep and monitors the heart rate of the wearer. If the heart rate increases, which can happen when the person is having a bad dream, it alerts the user through acoustic or tactile feedback to wake him up. (47 words)	*a smart watch capable of monitoring heart rate and determining that the wearer is asleep. It alerts user through acoustic or tactile feedback to wake him up when his heart rate increases due to a bad dream.* (37 words)	7
A wallpaper that has been coated with an ink that is sensitive to ambient temperature and changes its colour with temperature changes. (22 words)	*A wallpaper coated with an ink, sensitive to ambient temperature, changes its colour with temperature changes.* (16 words)	6

Table 6.3: Example of Verbose Queries and their well-written form

It is evident from table 6.7 that a verbose query impacts the relevancy of the search results. In the first query, there are numerous unnecessary words like *which can happen when* and *is having a*. Similarly, in the second query, *that has been* and *that is*. They unnecessarily increase the length of the text. Also, when we used a verbose query on PQAI with a stern look, we observed that the results provided by a well-written form of the query seemed more relevant than the other. We also noticed that some of the results that were present in the set of the well-formed query and not in the verbose query were present at the position from $11 - 20$. However, not all of them were there. It indicates that a query's verbosity directly impacts the search engine's indexing module. Louis et al. [LN14] suggested an approach to handle verbosity in a text. Their approach involves two factors - content type and article length. They assumed that some content types must be included and should not be treated as fillers in a verbose text. They use a collection of concisely written new articles and learn

the relationships between the properties of the text and the length of the articles. Given both content type and length of the text, they analyze that deviation from concise style is reflected by a mismatch between content type and length. Such articles are called of lower quality. In contrast, those with a good relationship between the content type and length are of good quality.

6.1 Feature Extraction

Louis et al. [LN14] proposed some features for characterizing content types in a text. We use the same set of features for a model to predict the length of a query text. These features are not extracted from a whole query but from a query snippet. We will discuss snippet selection in the next section. We will extract 87 features from each query (snippet). Suppose Q is a set of queries, and for each query, we select a snippet of length k. A set with all the query snippets is $\bar{Q}$. For each snippet $\bar{q_i} \in \bar{Q}$, we extract the following features [LN14].

6.1.1 Length of Units (10 features)

The length of the unit feature set involves basic sentence and redundancy properties of a sentence. In this feature set, we include:

1. number of sentences
2. average length of sentences in words
3. average word length in characters
4. type-to-token ratio (TTR)

The TTR can be estimated using the following formula:

$$TTR = \frac{\text{Number of types}}{\text{Number of tokens}} \times 100$$

Additionally, we also extract the number of noun, verb, and preposition phrases and their average length in words.

6.1.2 Syntactic Realization (30 Features)

To extract syntactic realization features, we require sentences from an external dataset. We utilize 50000 random sentences taken from European Patent Office (EPO) patents for this set of features. We generate a parse tree for each sentence using Stanford Core NLP [Man+14] and compute their grammatical productions. Gathering all the grammatical productions together, based on the count of each production, we select the 15 most frequent productions whose LHS is a noun phrase (NP). We also select the 15 most frequent grammatical productions whose LHS is not a noun phrase. These features will help understand the model and the type of information attached to the query snippets. List of these productions is given in table 6.4 and 6.5.

NP → NP PP	NP → DT NN	NP → DT JJ NN
NP → NN	NP → CD	NP → DT NN NN
NP → NP VP	NP → NNS	NP → NP CC NP
NP → JJ NNS	NP → JJ NN	NP → NNP
NP → NP	NP → DT NNS	NP → DT JJ NN NN

Table 6.4: 15 most frequent Grammatical Productions with NP in LHS

PP →IN NP	. →'.'	DT →'the'
ROOT →S	DT →'a'	IN →'of '
CC →'and '	S →VP	S →NP VP
-RRB- →'-RRB-'	-LRB- →'-LRB-'	IN →'to '
ROOT →NP	IN →'in '	ADVP →RB

Table 6.5: 15 most frequent Grammatical Productions with non-NP on LHS

6.1.3 Discourse Relations (5 features)

Discourse relations features are based on the hypothesis that different discourse relations vary in their appropriateness for articles of different lengths [LN14]. We utilize a tool named *Add Discourse* by [PN09] that recognizes explicit discourse relations in the snippets of patent abstracts. The potential senses could be - comparison, expansion, contingency, and temporal. The tool is originally written in Java, and to use a java class in our python code, we use *jnius*[1]. We use the number of each four classes of these discourse relations in the abstract snippet sentences as features. Furthermore, we add the total number of discourse connectives in the snippet as features.

6.1.4 Continuity (6 features)

Continuity features help the model identify the information provided in the adjacent sentences. It is obvious that if the information in the adjacent sentence is related, it indicates the topic being discussed is also continued in the further sentence. Otherwise, it indicates a change of topic. Usually, in long articles, this pattern can be observed where a topic (or its subtopic) is discussed in multiple sentences. This information may help our model indicate the difference in the snippets.

We include the number of pronouns and determiners as features. Additionally, we include the average word overlap value between adjacent sentences. To compute this value, we use *CountVectorizer* [Ped+11] that provides a word vector containing the count of words among the snippet sentences for each sentence. Then we calculate the cosine similarity of each sentence with its adjacent sentence. We add the average of these cosine similarity values of a snippet as a feature. Cosine similarity between sentence A and sentence B can be calculated as:

$$similarity(A, B) = cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

We believe that coreference information would also help identify the continuity across sen-

[1] https://pypi.org/project/jnius/

tences. Coreference information helps identify expressions that direct to the same entity. Thus, it could be valuable information for these types of features. Using the Stanford Coreference Tool, we check for pronoun and entity coreference links present in a sentence or any other sentence of the snippet. We add the counts of intra-links, intra-links, and the total number of coreference links as features.

6.1.5 Amount of Details (7 features)

In this set of features, we use the predictions of our Query Specificity Model (QSM) 5 for all the sentences in the snippet. We calculate the percentage of specific (predicted) sentences in a snippet and the average specificity of words using the distance of the words to their root word using WordNet [Mil98]. We also add the number of descriptive words like adverbs and adjectives, named entities, the average length of named entities, and the number of sentences with no named entity as features.

6.1.6 Compression Likelihood

This set of features uses an external source of information. Compression likelihood features possess the type of unnecessary information that is often ignored while summarizing a document. Hence, we assume that a document summary contains the most helpful information. We retrieve a USPTO dataset having 12480 instances with title and their respective abstracts. We believe that a user of prior art search engines uses particular details while forming a query and does not explicitly include long sentences explaining a topic. Additionally, It would be a strong compression because so much of the information will be deleted when an abstract document is compressed to a title. However, we are aware of the fact that the query written by a user could be more significant than a title of a patent document. However, we expect an ideal query to focus on specifics like titles. Therefore, this dataset would give brief information on the rules and patterns that are ignored in forming titles.

We use the Stanford CoreNLP tool [Man+14] to get parse trees of the abstracts and their titles. We extract the grammatical productions of both using the parse tree. We determine all the productions from abstracts if they undergo deletion or not. A production (LHS $\rightarrow$ RHS) undergoes deletion if LHS or any node in RHS does not appear in the title sentence. Only non-terminals are considered for this analysis. The ratio of times a production undergoes deletion is known as *deletion probability*. We estimate *deletion score* to obtain a set of 25 most deleted productions. It can be computed using the following formula [LN14]:

$$\text{deletion score} = \text{deletion probability} \times \log(\text{frequency of production in abstract})$$

To extract features from the snippets, we indicate the occurrence of the 25 most deleted productions (shown in table 6.6) in the snippet. We also include the sum, average, and product of those productions' deletion probabilities (calculated from previous steps), which are also available in the snippet's set of productions. We also include *perplexity* calculated using the product of these probabilities.

S → NP VP .	ADVP → RB	PRN → -LRB- NP -RRB-	VP → VBZ NP
VP → VBZ VP	WHNP → WDT	NP → CD	VP → TO VP
S → VP .	SBAR → WHNP S	NP → NP SBAR	VP → VBN PP
VP → MD VP	ADVP → IN RBS	WHADVP → WRB	VP → VBP VP
PP → IN NP .	QP → ADVP CD	VP → VBZ PP	ROOT → SBAR
NP → NP PP .	NP → DT JJ NN	SBAR → WHADVP S	ROOT → FRAG
S → CC NP VP .			

Table 6.6: 25 most deleted Grammatical Productions

6.2 Snippet Selection

We select a chunk of text from the abstract documents of BIGPATENT [SLW19] dataset of size $k = 50$. The minimum length of abstracts in our is 14, below the threshold. Therefore, we omit the examples of lengths below 50. The maximum length of an abstract from the dataset is 403. We ensure that the snippet selected includes whole sentences, and we do not strictly cut the sentence to follow the threshold of 50. If the sentence exceeds the threshold by 20 words, we allow it. If it crosses the 70-word limit, we omit the example for further process. The snippets are selected separately from the abstracts' beginning and end. The features explained above are extracted from START and END snippets separately.

6.3 Data

We gathered snippets from 11248 abstract documents of BIGPATENT [SLW19] from START and END for the training set. We have extracted 87 features for each snippet, which are kept separately for both types. Likewise, we have 633 and 629 snippets in the test and validation set, respectively, for both varieties of snippets. Thus, we have feature sets of sizes $[11248 \times 87]$, $[633 \times 87]$ and $[629 \times 87]$ for training, test and validation respectively. Since the task is to predict the lengths of each abstract document, the ground truth is the actual sizes of the documents.

Snippets ($k \simeq 50$)	**Length**
dispensing within a liquid dispensing and suctioning system is controlled by pinching a flexible tube with a spring - biased actuator. the system includes a slidable extension for pulling the actuator to open the tube to dispense liquid to a surface to be cleaned. the system is assembled with a tubular wand of a wet / dry suctioning system.	124
a surgical plate and process for preventing screw backout of repaired bones. at least one pawl is provided on a surgical plate adjacent to a screw hole. a screw having a ratchet wheel is inserted through the hole and screwed into the bone. the pawl engages the ratchet wheel to prevent rotational movement of the screw to prevent the screw from backing out.	123

Table 6.7: Snapshot of the Dataset used for training verbosity model. Left column shows snippets of abstract from BIGPATENT and right column has length of abstracts.

6.4 Verbosity Detection Model

We employ two types of length prediction approaches in this thesis. First, a classification model predicts the appropriate range of length of the snippets, and another regression model predicts the actual lengths. However, the approach to building both models is similar; only the labels change. Suppose we have set of eligible abstract documents D of size n such that $D = \{d_1, d_2, d_3, ...d_n\}$. Using a length threshold k, two sets of snippets S_{START} and S_{END} is selected i.e. $S_{START} = \{s_1, s_2, s_3, ...s_n\}$ and $S_{END} = \{s'_1, s'_2, s'_3, ...s'_n\}$. The set of actual lengths for START snippets can be denoted as $l(S_{START})$. The task is to predict the length $\hat{l}$ of the original documents of each snippet.

6.4.1 Length Range Classification

We distribute the lengths of abstract documents in 4 groups shown in Table 6.8. We test various machine learning and deep learning methods to predict the range. These methods, with their parameters, are described below.

Length Range	Number of Documents
0-100	4742
100-200	6059
200-300	440
300-400	7
400+	1

Table 6.8: Number of documents for each length range

6.4.1.1 Deep Learning Method

We create a two-layered simple neural network model. The model has two linear layers, each followed by a ReLU activation function and the last layer is a Softmax Layer. We calculate Cross Entropy loss to estimate the gradients. We train this model in batches of size 16 with a learning rate of 0.0001.

6.4.1.2 Support Vector Machine

Using the approach followed by Louis et al. [LN14], we train a Support Vector Machine (SVM) model with the radial kernel. We train another model with a linear kernel also.

6.4.1.3 Random Forest Classifier

The Random Forest Classifier is best suited for our data. We create a pipeline of methods to be applied to our data. We select the best 56 features from our feature set. We came up with this number of 56 because the P-value of other features crosses the threshold value of 0.05.

In the Result section, we will report the P-values of the best and worst features. Therefore, we ignore them while training the model. We select these features using Chi-Squared scores of each feature. Our features are non-negative; therefore, our Chi-Squared score best suits the task. We normalize the data using StandardScaler [Ped+11] that transforms data such that its distribution has a mean value of 0 and a standard deviation of 1. For a given feature, subtract the mean value from each point and divide it by the standard deviation. The third component of the pipeline is the RandomForestClassifier [Ped+11] itself. We check multiple maximum depths of the tree, starting from 2 to 20. The classifier with the highest accuracy will be discussed in the results section. We choose Gini Index (formula given below) as the splitting criteria for an attribute. The rest of the parameters are default. For example, the minimum number of samples required to split an internal node is 2, the minimum number of samples required to be at a leaf node is 1 and *sqrt* to compute the number of features to consider when looking for the best split [2].

6.4.2 Length Prediction

The length prediction model is different from the previous model in that here, we will predict the lengths of the articles based on their snippet's features instead of predicting the length range. We use the same dataset that we used for the length range prediction. However, this model requires some extra steps that we will briefly discuss in this section. There is a total of 11249 abstract documents in our training dataset. Each article's abstract describes an invention in a very concise way. Some abstracts are longer than others, and some are very short. We believe each abstract does not contain the same topic's information in the entire document and gets changed after two sentences. Let us understand it with an example given below.

> a handheld navigation device for use by the visually impaired having a camera electrically connected to a microprocessor. the microprocessor is capable of object and character recognition and translation into braille. a braille display is electrically connected to the microprocessor. a speaker is electrically connected to the microprocessor for audibly communicating common objects and distances and character recognition translations to the user.

In the example, we look at the first two sentences, they are explaining the same topic. The second sentence can be assumed as the extension of the first sentence. Similarly, there is mention of the topic in the third sentence and the second sentence. However, the fourth sentence is slightly off-topic from the first three sentences. Although it should not be clearly noticeable for a human, it may cause a difference for a machine. Only the word *microprocessor* is available in the previous three sentences in the fourth sentence. As a result, if we calculate the cosine similarity between the four sentences from the first three, it would be lower. At least, it would be lower as much as it can not be considered to belong to the same topic. Therefore, we directly use this data without segregating the topic from a document individually, we believe that features generated in the previous section would not properly represent the document's content, and it may end up confusing the model. As a result, we get poor predictions from the regression model. To avoid this situation, we perform topic segmentation on each abstract document.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

6.4.2.1 Topic Segmentation

Louis et al. [LN14] uses a different approach to topic segmentation to extract topics from their human written summaries of news article dataset for their length prediction model. However, we believe that their approach might not be suitable for our documents from the patent domain. Therefore, we inspire by the state-of-the-art approach to topic segmentation proposed by Solbiati et al. [Sol+21]. Their approach is an unsupervised method that uses a pretrained model called Roberta and computes the embeddings for each sentence. However, their method is used on transcripts of online meetings where the data is time series. Also, the task is to check if there is a topic change between two sentences. Our approach requires one extra step of segregating the sentences that do not belong to the same topic. Hence, forming groups of sentences for each topic.

Firstly, we select whole abstract documents from the dataset and break them into sentences. In the BIGPATENT [SLW19] dataset, it is very common that long sentences are separated by commas instead of full stops. So we identify such punctuations and replace them with full stops. However, in some cases, we will replace commas with a full stop where it is not needed but will be handled in the end result when we join the sentences. Then we split each article into sentences. We use BERT-for-Patents, a pretrained model, to compute the vector representation of each sentence in the article. These embeddings are extracted from the second last hidden layer of the pretrained model [Sol+21]. This step gives embeddings of each token. Then the vector representations of the tokens are fed to an average max pooling layer to compute the sentence embedding. For instance, if a sentence is ten words long, the pretrained model returns a matrix of 12×1024, including [CLS] and [SEP] token's embeddings. After average pooling, it becomes of size $1 \times n$ where n is not fixed but is less than 1024, and it varies for every sentence embedding. The flow of this method is shown in figure 6.1.

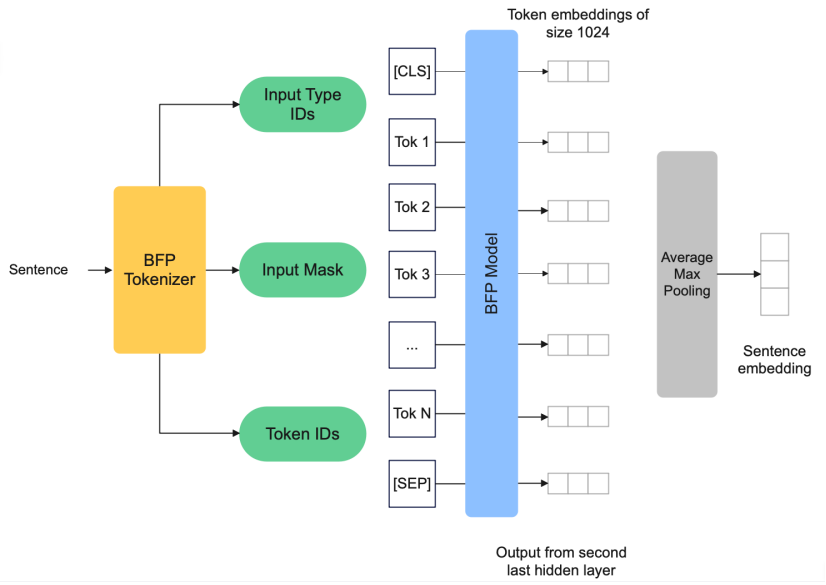

Figure 6.1: Sentence encoding using BERT-for-Patents (BFP) model.

Since the size of sentence embeddings of each sentence is different, we make all the sentence's embeddings size equal to the sentence embedding of maximum size. In the last step, we compute the cosine similarity of each sentence with each sentence in the article. This step gives a symmetric matrix, $sim\_mat \in \mathbb{R}^{m \times m}$, where m is the number of sentences in the article.

To identify the sentences belonging to the same topic, we apply the k-means algorithm to form clusters of the topics. To apply the algorithm, we should have at least 3 sentences in the article if we expect it to make at least two clusters. therefore, we omit the articles if they do not have a minimum of three sentences. We want to make sure that one topic at least contains 2 sentences. Hence, the number of clusters is calculated by the number of sentences in the article divided by the required length, three. Based on the k-means algorithm's prediction, we join each cluster's sentences and consider them a unique topic segment. After this step, among 11249 articles, we have 21164 topic segments ranging from 18 to 240. Since we have decided on the snippet size of 50 in the previous model for features extract, we also want to keep the same size for the topic segments. Therefore topics segments below the size 60 are omitted from the set. Finally, we have a set of 9029 topic segments ranging from 60 words to 240 words.

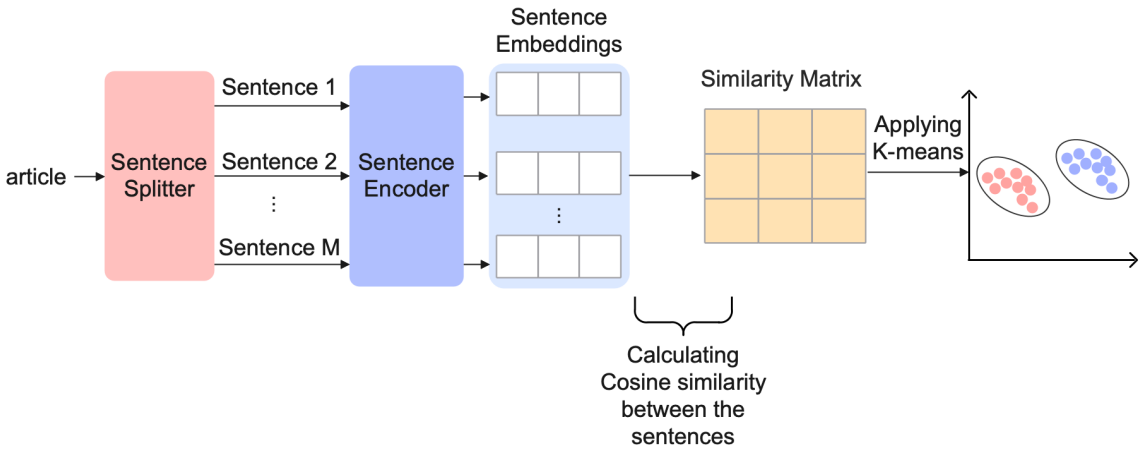

Figure 6.2: Topic Segmentation

We divide these 9029 topic segments into bins. These bins are created for every 30-word increase in the number of words of the segments, i.e., for the segments of length 60 to 240, we create 6 bins. Most of the topic segments belong to the first bin in the range 60 to 90 and for the subsequent bins number of segment decrease. The last bin of range 210 to 240 has only 9 topic segments. A complete distribution of topic segments into the length range is shown in the Table 6.9.

We use a stratified sampling method to train the regression model to ensure that the training and test set contains topic segments from each bin. The proportion of segments selected is based on the number of documents in the bin.

Length Range	Topic Segments
60-90	5041
91-120	2725
121-150	994
151-180	226
181-210	34
211-240	9

Table 6.9: Number of topic segments for each length range

6.4.2.2 Length Prediction Model

As we did in our previous model, we extracted features discussed above for the topic segments. The length of snippets is kept at 50 for this method, also. In the training set, we have 7224 segments in the training set while 1805 in the test set. For each topic segment, we collected 87 features. Therefore the size of the training set is 7224×87, and the test set is 1805×87. We train a linear regression model with ordinal least squares using a python library Statsmodel[3]. We use the actual length of the topic segments as labels or the dependent variable. The features that turned out to be most and least significant are indicated by standardized coefficients (Beta) (in Table 6.10) associated with a t-test to determine if a feature can be ignored from the regression model.

Positive Coefficients		
Feature	**Coefficient**	**P-Value**
Avg. Word Length	55.2220	***
Avg. Deletion Prob.	40.8913	***
Product Deletion Prob.	7.3096	***
Avg. NP Length	4.4254	***
Avg. Word Specificity	1.2003	***
NP Counts	0.8104	***
Negative Coefficients		
Product Deletion Prob.	-16.8936	.
Avg. Sentence Similarity	-13.9676	***
Number of Sentences	-9.9844	***
VP → MD VP	-2.3370	***
SBAR → WHADVP S	-1.9198	.
ADVP → IN RBS	-1.8406	**

Table 6.10: Most and least Significant Features with their Standard Coefficients. '***' indicates P-value ≤ 0.05, '**' indicates p-value > 0.05 and '.' indicates P-value ≥ 0.1

[3]https://www.statsmodels.org/dev/generated/statsmodels.regression.linear_model.OLS.html

7 Results

So far, we have covered a variety of in-depth techniques for identifying errors in a natural language query for prior art search, as well as how to put them into practice. In this section, we will go through each method's performance in detail and assess how it stacks up against other approaches. This section focuses on locating a query's errors and evaluating the applicability of the results. We will check to determine if a perfectly written query has higher relevancy to the search results.

7.1 Grammatical Error Detection

For grammatical error identification, we have trained two models. We will discuss their performance individually. The Pytorch library (version 1.12.0) and Cuda 11.3 were used to train each model in this section on an Nvidia Tesla T4 GPU.

7.1.1 Coarse-grained Classification

In this method, we fine-tuned a pretrained BERT model (called Bert-for-Patents) with the data we created using various errors people make while composing a sentence. We performed an error detection process to identify whether a text is grammatically correct or not instead of detecting individual errors. Hence, this becomes a binary classification task. As discussed in the methodology, we adjust multiple parameters like batch size, learning rate, and the number of epochs to get the best-fitted model. We track the model's performance on the validation set. Figure 7.1 shows the accuracy scores on validation set attained by the model discussed in 4 for various parameters.

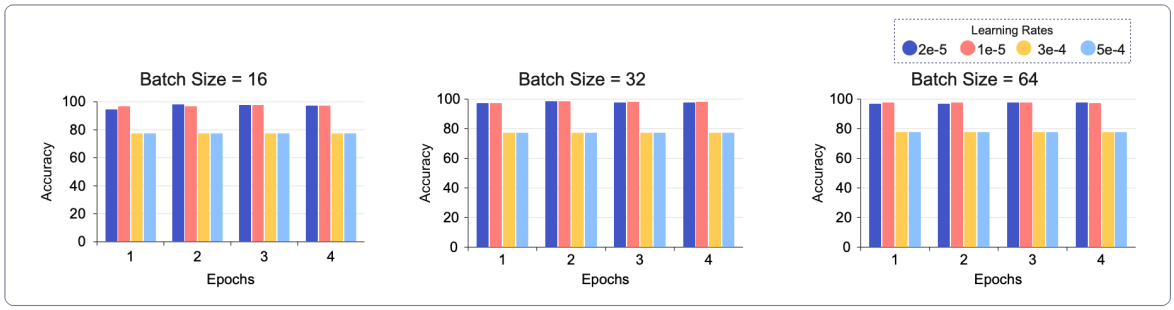

Figure 7.1: Accuracy scores on validation set achieved by Coarse-grained Classification model

Figure 7.1 shows that for every batch size, the accuracy score for learning rates 3e-4 and 5e-4 is worse. Additionally, the binary cross entropy loss is higher with these learning rates. As a

result, we cannot use these two numbers as learning rates when developing our model. The model is improved and receives better scores when the other two learning rates, 2e-5 and 1e-5, are used. When trained for two epochs, with a batch size of 32 or 64 and a learning rate of 1e-5, our model performs best on the two epochs. The model's accuracy scores in these two instances are 98.10 percent and 97.59 percent, respectively, with losses of 0.08 in each case. Nevertheless, the accuracy for batch size 32 with a learning rate of 1e-5 is somewhat better; for this reason, we select this set of parameters for further evaluation on the test set. Overfitting could result from training for more than two epochs, which increases the loss on the validation set.

We also make predictions on the test set using the model that performs the best on the validation set. Our evaluations are reported using the F1 score. In order to evaluate the effectiveness of our model, we have additionally trained a baseline. The test set contains 697 documents, of which 539 are grammatically correct and 158 are incorrect. Since our dataset has an imbalanced amount of labels thus, we report the weighted average of precision, recall, and F1 scores. Our model and the test set baseline display these results in Table 7.1.

Model	Precision	Recall	F1-Score
GED (CG)	**0.98**	**0.98**	**0.98**
Baseline	0.82	0.72	0.76

Table 7.1: Evaluation Scores on Grammatical Error Detection (GED) model and the baseline

From the confusion matrix shown in Figure 7.2, our model correctly predicts 148 documents to be incorrect out of 158. Only 10 of them were incorrectly classified as grammatically correct. However, only five documents were predicted as grammatically incorrect while they were correct.

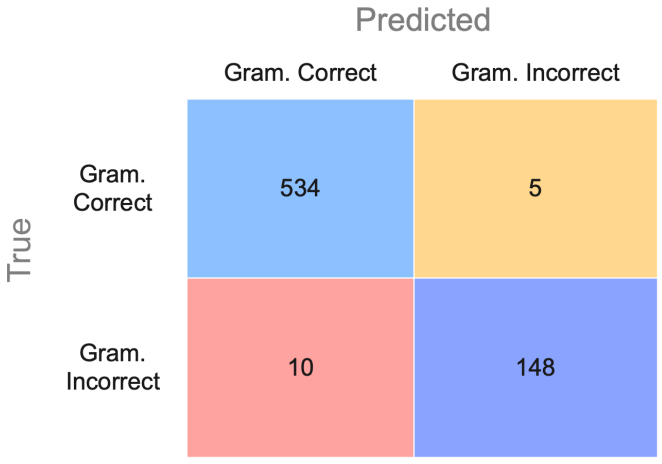

Figure 7.2: GED-CG classification model Confusion Matrix

About 6.75 percent of the findings in the confusion matrix are false positives. We suspect this might be because our model was trained on a dataset with a significant class imbalance, meaning that the training sample only contained 23% items with grammatical errors. Additionally,

we think this may have improved if our training dataset contained more incorrect samples. However, our overall approach has been practical and would work better with datasets of a similar size.

7.1.2 Fine-Grained Classification

We finetuned BERT-for-patents pretrained model to perform fine-grained classification using the training dataset. We used the same dataset for training the coarse-grained classification model. Merely the labels in this method are changed. Instead of classifying whether the document is grammatically incorrect or not, we are determining whether the incorrect document has specific types of errors. In chapter 4, we briefly discussed types of 5 types of errors introduced in the dataset. Therefore, we must predict a set of 1×5 labels for each document. Like the method mentioned above, we tune the model's parameters like batch size, learning rate, and the number of epochs. While tuning parameters for the previous classification model, we observed that the learning rates $3e-4$ and $5e-4$ did not enhance the model's performance. Therefore, we will disregard them for this model. We believe that these learning rates will not improve this model also because this model is very comparable to the previous one. The rest of the parameters were kept the same as the previous method. Accuracy scores on the validation set by the model trained with various parameter settings were recorded. Figure 7.3 shows the results.

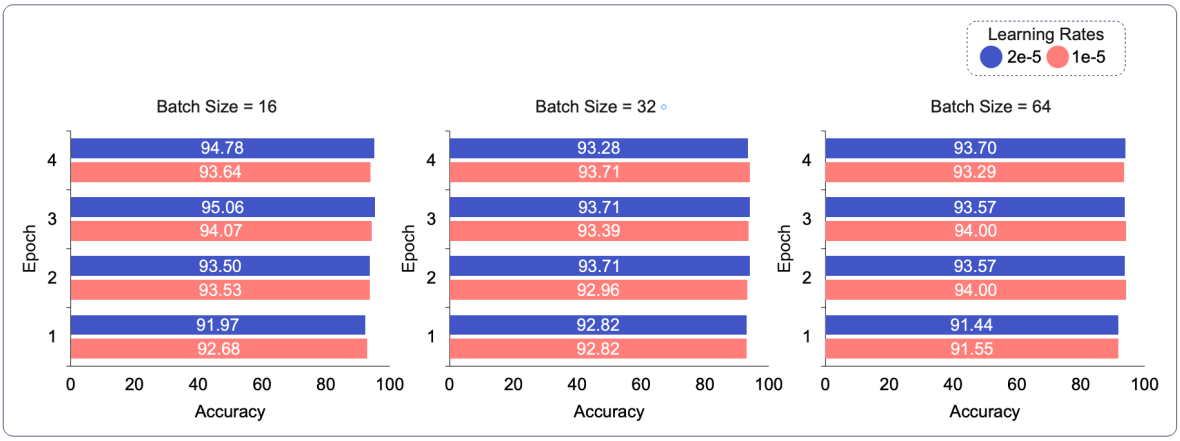

Figure 7.3: Accuracy scores on validation set achieved by Fine-grained Classification model

We also report the cross entropy loss value at each epoch in Figure 7.4. The model records the highest accuracy, i.e., 95.06%, when our GED-FG classification model was trained for three epochs with a batch size of 16 and the learning rate was $2e-5$. The second best accuracy, i.e., 94.78%, is achieved while all parameters were the same, but the model was trained for 4 epochs. Although we record a better score at the third epoch when compared to the fourth, the cross entropy loss is negligibly higher than on the fourth. Therefore, a tradeoff between the accuracy score and loss can be observed. However, while choosing the best parameter setting for our model, we would prefer the one giving a higher score because the difference in

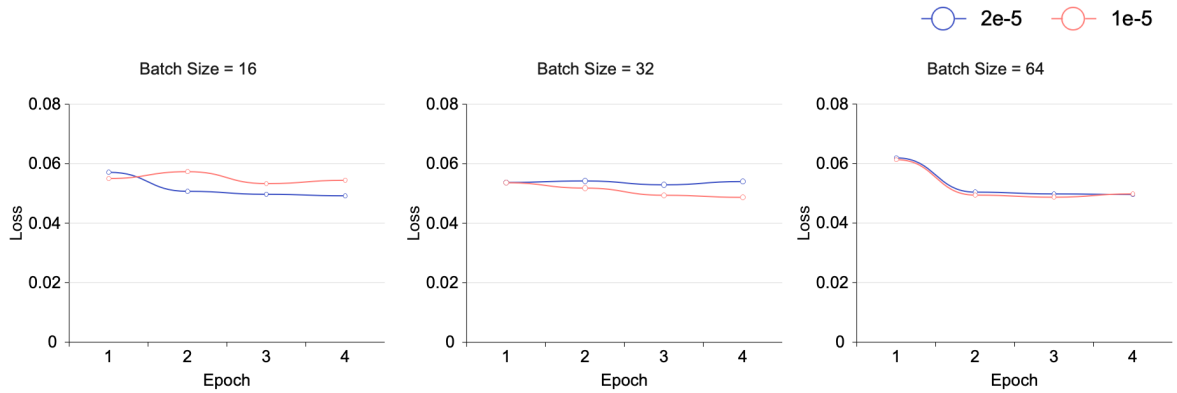

Figure 7.4: Loss on validation set

loss is negligible and could be ignored. Thus, we select the batch size to be 16, the learning rate is $2e-5$, and the number of epochs to be 3.

Now we make predictions on the test set using the model trained with the best parameter setting. Although, it should be noted that a sample from the data could have multiple labels; hence it is a multi-label classification task. Therefore, only precision-recall and f1-score may not be enough to evaluate the model's performance. The model could predict entirely true or partially true results in such cases. Therefore, in addition to the evaluation measures mentioned earlier, we report the exact match ratio, Jaccard score, and hamming loss for the test set prediction. All these scores on the test set are given in Table 7.2.

Metrics	Score (%)
Accuracy	93.97
Precision (Micro)	94.00
Recall (Micro)	89.00
F1-Score (Micro)	91.00
Exact-Match Ratio	93.97
Jaccard Score	84.17
Hamming Loss	0.01262

Table 7.2: Evaluation Scores on GED-FG model

Out of 697 samples in the test set, 655 were predicted correctly. That means the model prediction was entirely the same as the true labels. On the other hand, a very low hamming loss tells us that On the other predictions that do not entirely match with the true labels, the model was able to indicate some of the errors in the same. However, it is crucial to dig deeper into the results and analyze the model's performance for each class with the help of confusion matrices of each class, shown in Figure 7.5.

Looking at each class's confusion matrices, we notice that the model best differentiates spelling errors. There is only one article where it was unable to identify a spelling error and one article where it incorrectly points out a spelling error. There is a possibility that the word that was

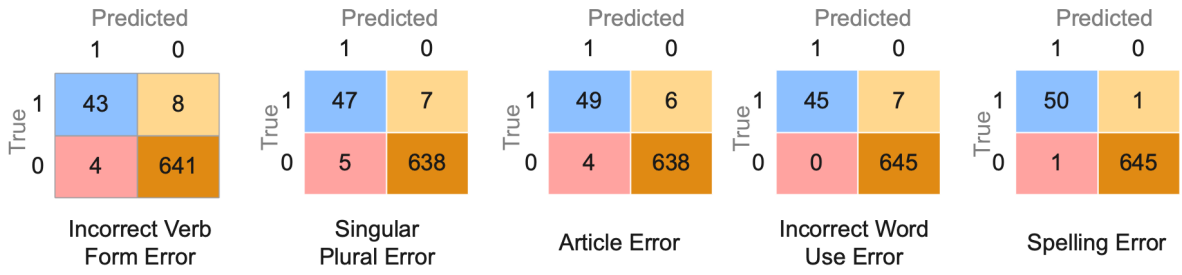

Figure 7.5: Confusion matrix for each error type

incorrectly identified with having an error could be a homonym of another word. However, the model struggles a little in identifying the *incorrect verb form* and *singular/plural* errors. However, we cannot deny that the model was trained with only 25% of the error-containing articles, and the performance could have been better if we had more data. Looking at the imbalance in the training data, the model still performs sufficiently well.

7.2 Query Specificity Model (QSM)

We construct three classifiers to distinguish between general and specific texts. While the second is trained using Specification sentences, the first is trained with Instantiation sentences. Both datasets are combined to train the third model. In this data, we noticed a link between the sentences where the first statement serves as an overview of the following statement. More specific facts are included in the second sentence. Hence, we treat the first statement as general and the subsequent statement as specific. We break the examples into two types of sentences and label them individually. We do not preserve any pairing information of the statements. The dataset is balanced in all these forms, with an equal amount of general and Specification sentences. We employed the logistic regression approach to categorize these two sentences. According to [LN11], probability-based predictions would be more appropriate to link with each statement rather than a strict classification of the two classes. Therefore, logistic regression would be more suited for this task. To estimate the performance of our models, we performed 10-*fold cross validation* for all three models. The minimum, maximum and average scores of cross-validation for Instantiation and Specification data are reported in Table 7.3.

Dataset	Min. Accuracy	Max. Accuracy	Avg. Accuracy
Instantiation	72.15 %	**84.25** %	78.87%
Specification	62.38%	67.99 %	65.14%
Instantiation + Specification	66.71%	70.96%	67.90%

Table 7.3: 10-fold Cross Validation scores of Specificity Model

The third model was trained using both types of texts in the hopes that doing so would improve the model since the training set was twice as big as the separate sets. But this assumption was incorrect. This model was able to outperform the model trained with just

Specification sentences in terms of cross-validation scores, but it was unable to outperform the model trained with Instantiation sentences. Consequently, we will use the best model (trained with Instantiation text only) to calculate more findings.

7.2.1 Feature Analysis

In order to train our model, we have incorporated both lexical and non-lexical characteristics. Except for word features, all of the features addressed in 5 are non-lexical. To determine which elements are more crucial for categorizing general and specific sentences, we trained a logistic regression model using each type of feature individually. This led us to conclude that non-lexical characteristics are crucial in this categorization. We saw better results from the model trained exclusively with the non-lexical characteristics than the lexical features. Furthermore, we see that the lexical characteristics only add 2% to the model's total accuracy. Table 7.4 provides a summary of all these results.

Features	**Accuracy**
Sentence Length Features	68.56%
Polarity Features	71.31%
Specificity Features	56.97%
NE+CD Features	74.06%
Syntactic Features	68.56%
Language Model Features	50.88%
Non-lexical Features	78.00%
Word (lexical)Features	72.10%
Total Accuracy	79.96%

Table 7.4: Feature Analysis (on a random split)

Given that the dataset contains an equal amount of general and specialized statements, the random baseline accuracy is 50%. We see that NE+CD features are the strongest among all the feature sets, followed by polarity features, whilst Language model characteristics and specificity are the weakest, offering just 0.88 and 6.97% improvement in accuracy over baseline, respectively. There is no discernible improvement for the model trained using Specification sentences. The language model features are similarly the poorest in this situation, whereas the NE+CD features are the greatest with an improvement over the baseline of 11%. On a random split, the accuracy with lexical features is 61.33%, while the accuracy overall with lexical and non-lexical characteristics is 64.25%. Similar trends can be seen in the combined version of the dataset, where NE+CD characteristics are the strongest and language model features are the lowest. Although the Instantiation dataset features scores surpass the other two, the results by this model are somewhat better than the scores by the Specification dataset features.

We chose a sample set of 40 sentences, 20 of which were general and the remaining 20 specific. We looked at how each feature contributed to the classification of general and specific texts. Even though some characteristics on their own were powerful enough to reveal the underlying patterns, others required additional features to discriminate between the labels. Here, it

was noted that the specific sentences had more words than the generic ones. We found that 75% of the general statements in the samples were up to 20 words long, whereas 76% of the specific sentences were longer than 20 words. In a similar vein, the polarity characteristics showed that there were more polarity words than sentences. Here, we observed that, when standardized by length, general statements contained 86% more polar terms. Similar to this, there were more identified named entities in the specific sentences.

7.2.2 Classification Evaluation

Up till now, we have examined each feature separately and provided the cross-validation score for each of the three models. Using the Instantiation model, we make predictions in this part based on the test dataset. We select 10% sentences at random for our model to predict from a sample of 2542 sentences. We have 127 specific sentences and 128 general sentences in our batch of 255 sentences. According to the confusion matrix Figure 7.6, out of 127 specific sentences, 100 of them were correctly identified by the model, and 97 general sentences were accurately classified as well. Here, 0 denotes abstract statements while 1 denotes concrete sentences.

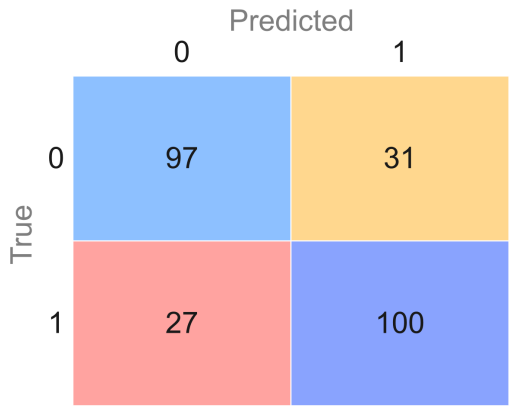

Figure 7.6: QSM-Instantiation Confusion matrix

The model reports 77.25% accuracy. Here, we evaluate the model further by computing, precision, and recall scores. We compute the precision and recall by macro averaging since the labels are balanced. The f1-score measurement is also reported. The Table 7.5 displays each of these scores.

Model	Precision	Recall	F1-Score
QSM-INS	76.33%	78.74	77.51%

Table 7.5: Evaluation Scores for QSM

Our specificity model predicts the likelihood that a query sentence will be either general or specific. The default threshold value for interpreting the probability was set at 0.5. However, in order to decrease the number of false positives and false negatives, it is crucial to record

predictions using different threshold values. The Receiver Operating Characteristic (ROC) curve is a helpful tool for interpreting the probability of a binary event. The false positive rate (FPR) and true positive rate (TPR) are compared for a range of possible threshold values between 0.0 and 1.0 on this curve. Therefore, given the probability predictions of our model, we display a ROC-curve 7.7.

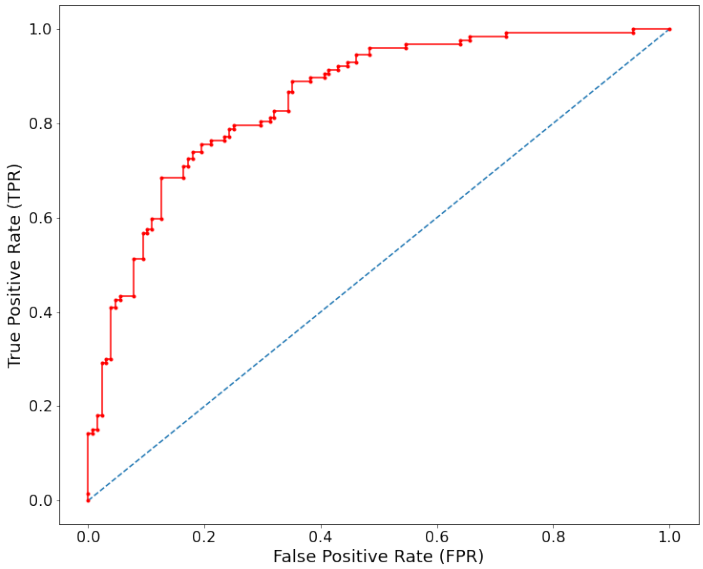

Figure 7.7: ROC Curve for QSM-INS

The Area under the Curve (AUC) value of 85.33% reported by our model shows that our classifier is capable of differentiating between general and specific texts. We may deduce from the ROC curve in Figure 7.7 that threshold 0.6 is preferable for our classification and threshold 0.5 is not the best value for comprehending the class labels. As a result, we selected the best threshold value for our subsequent estimations.

7.2.3 Annotation of Prior art Search Queries

The fundamental goal of this implementation was to annotate each sentence of prior art search queries expressed in natural language. A prior art query, in our perspective, must be precise and contain more specific words than general text. A prior art query often starts with a general first and (or) second sentence and then discusses the first and second sentences in more detail in the other sentences. The field of natural language queries from the state-of-the-art has seen far less study. It would be challenging to locate a dataset with prior art queries because the search engines created for this purpose do not gather them due to the confidentiality of the inventor's notion. As a result, we need to develop a new method for extracting queries. We searched through a number of previously filed patents to find statements that fundamentally describe innovation and use them as prior art search queries.

Mark your choice with ☑			
	Query	**Gen.**	**Sp.**
1.	The textile fabrics made from the present yarn have a smooth surface.	☐	☐
2.	Smooth surface enabling the assembly to be picked up and placed using automatic equipment with a vision system.	☐	☐
3.	The print apparatus further includes a laser beam scanner and a beam detector for producing a scan position signal.	☐	☐
4.	The actuator rotationally couples another of the rotational members of the second planetary gear set to the differential.	☐	☐
5.	Various forms of vertical tensioning support can be provided to include soil nails.	☐	☐
6.	The mounting nut includes a first segment and a second segment attached by a hinge portion.	☐	☐
7.	Seller automatically receives notification by one or more electronic methods such as an internet communication or a telecommunication text message.	☐	☐
8.	The invention also concerns the specific design of a windmill with such a slip generator.	☐	☐
9.	Actuation of the restrictor restricts the flow of drilling fluid from the drilling fluid passageway into the well environment.	☐	☐
10.	The outer form thermally insulates around the periphery of a poured concrete slab.	☐	☐
11.	A NAND flash memory device incorporates a unique booster plate design.	☐	☐
12.	The housing also includes an adjustment element pivotally mounted in the housing wall.	☐	☐
13.	An escrow service is provided by the business model should clients prefer a secure transaction.	☐	☐
14.	and one of the signals indicative of the slowest rear wheel speed and the front wheel speeds.	☐	☐
15.	The essential oils derived from renewable sources such as for example biodiesel.	☐	☐
16.	One or more video overlay engines read graphics objects from the dram.	☐	☐
17.	The sensor may also control electrical energy for heating the sole plate.	☐	☐
18.	This dual - path processing may be performed on a bed of fluidized powdered material including a powdered metal material and a powdered flux material.	☐	☐
19.	The method first transmits a response request message from the PCF to identify the status of two or more PDSNS in the system.	☐	☐
20.	The combining algorithm takes the form of a weighted summation of the spatial data stream and temporal data stream inputs with the weights being a function of the relative velocity errors.	☐	☐

Table 7.6: General/Specific Sentence Annotation Survey

We gathered 20 sentences from patent filings that were not included in the model training to produce a gold standard dataset of queries. In a poll we performed, participants had to categorize each query expression as either general or specific depending on its properties. Four survey respondents with a high English language proficiency provided their responses. The survey resembled Table 7.6.

We compiled the replies from the survey respondents. No ground truth label existed for these statements. Therefore, we had to completely rely on what the participants decided. This study tested our classifier's ability to distinguish between general and specific texts that belong to the patent domain. To assess our model's efficiency, we divided the participants' choices into three categories. In one section, 80%-100% participants provided the same response. Additionally, it's conceivable that all of them reject the classifier's prediction. In the second section, 60%-80% participants share the same answers, and in the third, 50%-60% individuals have the same answers. These responses fall into three categories: ***strong***, ***moderate***, and ***weak agreement/disagreement***, respectively. These responses are shown in Table 7.7.

Annotator Decision	No. of Samples
Strong Agreement	9
Strong Disagreement	2
Moderate Agreement	1
Moderate Disagreement	1
Weak Agreement	5
Weak Disagreement	2

Table 7.7: Annotator's collective Agreements/Disagreements on classifier decisions

We use the majority vote method to interpret the survey replies. We concur with the participants' classification for replies that show a high level of agreement or disagreement. We agree with the designation that a majority has approved of the participants in the situation of moderate agreement/disagreement. Twenty sentences from the questionnaire were anticipated, with 10 receiving high or moderate agreement, and three receiving disagreement from the participants. Five samples were weakly agreed by the participants. As a result, the model's accuracy of 75% is close to the prediction accuracy on a random split of Instantiation sentences. As a result, we can say that our model is effective at determining whether a sentence is general or specific in a patent document.

7.3 Query Verbosity Model (QVM)

We adapted the text verbosity model discussed in [LN14] for detecting verbosity in the patent query texts. This model uses a set of 87 features of snippets of abstract texts from BIGPATENT [SLW19] dataset of length $k = 50$. All these features are briefly discussed in chapter 6. The goal of this model is to predict the length of a text from where the snippet has been selected based on the content characteristics of the document. To predict the lengths of documents, we trained two types of models, the first, which predicts the length range, and the second which predicts the actual length. For predicting the length range of the documents, we tried and tested three methods, a two feed-forward layer neural network model, a Support Vector Machine Model (SVM) with radial and linear kernel, and a Random Forest Classifier. For the regression-based length prediction model, we used trained a linear regression model. In order to determine if a text from the test set is verbose, we follow the approach adopted by Louis et. al [LN14]. Suppose a test set $D_{test} = \{d_1, d_2, d_3, ..., d_n\}$ with n documents. Using the snippet length of k, the snippets of documents are $S_{test} = \{s_{d_1}, s_{d_2}, s_{d_3}, ..., s_{d_n}\}$ and the

lengths of documents can be denoted as $l_{test} = \{l_1, l_2, l_3, ..., l_n\}$. The task is to predicts the lengths of snippets i.e., $\hat{l} = f(S_{test})$. The verbosity using the predicted length of a snippet can be interpreted as follows:

1. $\hat{l} \gg l$**:** means that the query document $d_n \in D_{test}$ is not concisely written and contains unnecessary information. Hence, it is verbose.
2. $\hat{l} \ll l$: It means that the predicted length indicates that the length of the query is too short of describing an invention. The query text lacks the necessary information that has to be included.
3. $\hat{l} \simeq l$: The predicted length indicates that the query text is concisely written and therefore is not verbose.

The following scores are calculated for quantifying the verbosity in a query.

Verbosity Degree : This score is the difference between the predicted length and the actual length of the document. A positive value of verbosity degree indicates verbosity, whereas a negative score indicates that the text is short [LN14].

$$\boxed{\text{Verbosity Degree} = \hat{l} - l} \tag{7.1}$$

Deviation Score : This score indicates the magnitude of error in the text. Being short or verbose, both are problematic for the text.

$$\boxed{\text{Deviation Score} = |\hat{l} - l|}$$

7.3.1 Length Range Prediction

As discussed in the chapter 6, we collected a set of 87 features from the snippets of the documents. In total, we have 11249 documents, out of which most of the documents lie in a length range of 0 to 100 and 100 to 200. Hence the models were trained using a feature matrix of dimension 11249×87 and the labels of dimension 11249×4. We selected the best 56 features based on chi-square scores.

The deep learning model employs binary cross entropy loss to compute the loss between the true and predicted labels. As we were training the model, we discovered that, after training for 14 epochs, the loss value on the validation set was very high. On the other hand, it did not receive a good accuracy score on the validation set. However, the model proved excellent at detecting texts between 100 and 200 words in length but struggled to accurately predict the other classes. While we presume that there were insufficient examples for training for length ranges 200–300 and 300–400, it also performed poorly for length ranges 0-100.

We also trained a Support Vector machine (SVM) classifier with radial and linear basis kernel on our patent data. Louis et. al [LN14] uses SVM with radial kernel for the range classification. However, with our data, the radial kernel seemed to be giving better predictions than the linear kernal. The SVM classifier performs a little better than the deep learning method, but we believe that we can get better scores using other methods as well.

We used the Random Forest technique to train the third classifier. The chapter 6 covered every parameter needed to train this classifier. However, until we discovered the best parameters that suited our data, we kept adjusting the maximum depth of trees. This approach performs remarkably better than the model developed by Louis et al. [LN14] in classifying our data. However, we saw a similar pattern in our model as in Louis et al. [LN14], which showed that their classifier becomes confused for the length ranges of 200–300 and 300–400. Our classification model looked impressive for the 0-100 and 100-200 word abstracts but not for the rest. This results from the training dataset's dearth of articles on specific ranges.

At last, we train an XGBoost classifier with the same training data, and amongst all the methods we discussed earlier, it performs the best. This model was trained with default parameters first and was able to give satisfactory results. As we tune the parameters the model showed a little improvement. We started with a learning rate of 0.1 and a maximum depth of the trees to be 2. With these parameters, the model achieved an accuracy of 58.71% of accuracy. Later we reduced the learning rate to 0.3 and the fraction of features that are used to train each to 0.6. With these parameter sets, we achieved the best score from this model. However, we experimented with various parameter sets, including learning rates of 0.02 to 0.09, the proportion of features used to train each tree between 0.5 and 1.0, and the maximum depth of the tree between 2 and 10. However, the model produces the best results only with the settings mentioned above.

We had 535 abstracts in the test data set. Accuracy scores reported by all these three types of methods are shown in table 7.8.

Method	**Scores**
SVM-RBF	55.67
SVM-LIN	51.41
DNN	58.99
Random Forest	60.72
XGBoost	**61.76**

Table 7.8: Accuracy Scores by length prediction models

To make predictions on the test dataset, we use the best-fitted model for further evaluation. From Table 7.8, we can see that the random forest classifier performs best on our test data. Therefore, we record precision, recall, and F1 scores by this model. Since we observe a huge imbalance in our data, and this imbalance causes misclassification, we train the model once again only with articles with a length of 200.

Model	**Precision**	**Recall**	**F1-Score**
QVM-RP	61.84%	61.76%	61.80%

Table 7.9: Evaluation Scores for QVM-RP

Our XGBoost classifier was able to identify 144 and 234 documents of length range 0-100 and 100-200 respectively in the test set. However, there is a big proportion of false negative and false-positives for each class. The confusion matrix of our model's predictions on our test set

is shown in Figure 7.8

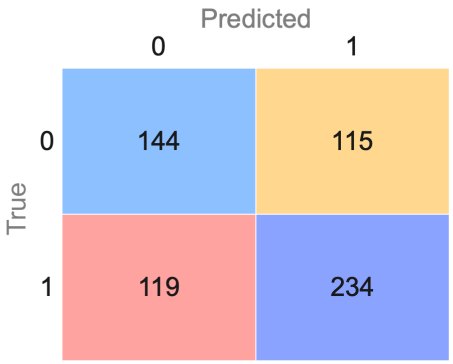

Figure 7.8: Confusion Matrix - QVM-LR

As we saw that the XGBoost classifier gives higher scores than all other methods we tried for length range classification. This shows a 6.09% increase in the score than Louis et al. [LN14] model for our data. However, we believe there is still much room for improvement, but our model's scores are satisfactory for research that has hardly any previous foundation.

7.3.2 Length Prediction

While the length range prediction has given satisfactory results on the data, we dig a little deeper to predict the lengths of the articles. This would help us determine the degree of verbosity present in the abstract documents. The length prediction model involved a few extra preprocessing steps before performing regression analysis where we segmented each article into topic segments. We trained an ordinal least squares linear regression model with 7224 topic segments with the length of topic segments as dependent variable.

On the test data, our model shows a significant improvement. The R^2-value of the model is 0.2047. We compute the correlation of the predicted labels with the true labels also which are shown in the Table 7.10. For all these three measure, we can observe the p-value is very small that indicates that the predicted lengths of the segments are highly correlated with the actual lengths. These results shows that our model successfully able to find a relationship between the content type and length of segments. Therefore, this model can be used for predicting verbosity in patent queries by computing verbosity degree and deviation score. We will compute these scores in the next section on patent queries.

Measure	**Score**	P-value
Pearson Correlation Coefficient	0.4545	8.775e-93
Spearman's rank correlation coefficient	0.4570	7.042e-94
Kendall rank correlation coefficient	0.3185	2.730e-90

Table 7.10: Length Prediction Model Evaluation Scores

8 Discussion

In this chapter, we will discuss the effectiveness of three of our models for the natural language prior art queries. These techniques will enable us to assess the usefulness of our models for AI-based prior art search engines.

8.1 Relevancy Estimation with PQAI Results

So far, we have developed three models to evaluate a patent query's or a patent text's overall quality. However, it is crucial to evaluate the outcomes of the operational search engine in light of the predictions made by these four models. Since none of the state-of-the-art search engines release the dataset of user queries in the public domain, the relevancy estimation begins by gathering queries. As a result, we created the queries by ourselves. Writing the queries manually would be very helpful because it would replicate a query created by an inventor, which would inevitably have issues with specificity and verbosity. However, we will explicitly add them if there are only a few errors in the queries we write.

Another strategy for gathering NL prior art queries is randomly selecting sentences from patents that only hint at the innovation rather than describing it thoroughly. Then, we rephrase our query text using Quillbot[1], a paraphrasing tool. The tool's advantage is that it modifies the non-lexical information while maintaining the sentence's lexical content. As they focus on the non-lexical aspects, this will be particularly useful in assessing the efficacy of our specificity and verbosity models.

8.1.1 Introducing Errors in the Queries

Our earlier presumption is that the queries are grammatically accurate because we obtained them using the two different methods we outlined in the previous sections. Therefore, using the same tool we used to construct the dataset for the GED model, we will insert grammatical errors in some of the queries with the error we explained in chapter 4. Because we assume that all queries would automatically contain general and specific sentences, we do not need to incorporate these sentences explicitly.

8.1.2 Query Dataset

A small dataset of 91 queries with grammatical errors and their well-written counterparts was collected. Using the Quillbot tool, we also generate an alternative paraphrase of these queries. We will run these queries and their appropriately well-written version through the

[1] https://quillbot.com/

PQAI search engine to determine the distance between the provided query and the search results.

8.1.3 Query Relevancy with the Results

In this thesis, we will compare the query we know has a specific problem to its respective well-written form that does not have that particular flaw in estimating the query relevancy on PQAI. For instance, if our GED model predicts that a query is grammatically incorrect, we will note the distance between that ill-formed query and the first result of its corresponding well-written version. Because the PQAI presents results in ascending order of similarity score, the first result is always the most relevant to the question. Therefore we decided to compare only the first result over the rest. Later, we compare the distances to check if there is an increase in the distance because of the error.

8.1.3.1 Relevancy of Queries having Grammatical Errors

We selected a few lines randomly from the query set and added grammatical errors. Our coarse-grained GED model predicted that 31 of the 91 patent-related questions would have grammatical errors. Initially, we run these 31 grammatical mistakes queries through the PQAI engine and log their distances from the first PQAI result. The distance between the first result and the well-written form of these four queries (without grammatical errors) is then calculated using PQAI. Some of these results are shown in the Table 8.1. In Table one on the left, we show the queries with specific types of grammatical errors we discussed in chapter 4. The table on the left lists the queries containing the specific grammatical faults we discussed in chapter 4. Examples include the first query's misspelled term and the second's incorrect word form. The third one has a word form error, several misspelled words, and an article mistake. We show the PQAI similarity score between the incorrect query and the first outcomes in the second column. The third column shows the queries from column 1 in their grammatically correct form. The fourth column shows the PQAI similarity score attained after the query was executed correctly. In all of the samples, we see that the score increased when the search engine was given a bad query and improved when the correct query was given. This shows that the search engine is sensitive to grammatical mistakes, and a grammatically accurate query helps users of the PQAI search engine or any AI-based prior art search engine obtain better results for their search.

8.1.4 Relevancy of Queries with Specific Sentences

We trained a logistic regression classifier that differentiates between general and specific sentences. We created two sentences to check the impact of specific information in the queries. One set only contains general sentences of each query, and the other contains both general and specific sentences predicted by our classifier. We record the scores of the first result from PQAI using both sets individually. In this test, we have equal size sets of 36 queries. In this test, we observed that only six times out of 36, only general sentences were enough to get relevant information from the search engine; otherwise, the specific information was helpful

Query with Grammatical Error	Score	Well-written form	Score
A powr drill that contains a battery but when the battery basically is discharged it can also mostly run on conventional fuel like kerosene, or so they definitely thought.	0.4410	*A power drill that contains a battery but when the battery basically is discharged it can also mostly run on conventional fuel like kerosene, or so they definitely thought.*	0.4092
An speech correction system that diagnosed the issues associated with the speech of a non-native speaker and then corrects the non-native accent in real time so that the corrected speech sounds like a native speaker's speech.	0.2127	*An speech correction system that diagnoses the issues associated with the speech of a non-native speaker and then corrects the non-native accent in real time so that the corrected speech sounds like a native speaker's speech.*	0.1800
A LED light bulb for use inn homes that adjusts its color temperature depended on the reflective properteis of the roomś wall, e.g., due to the paint.	0.2039	*An LED light bulb for use in homes that adjusts its color temperature depending on the reflective properties of the roomś wall, e.g., due to the paint.*	0.1983
an automatic focused camera according to the present invention included an automatic distant measuring system which generating a signal indicative of a distance betweeen the camera and a target object to be photographed upon receipt of electric powr.	0.1367	*an automatic focusing camera according to the present invention includes an automatic distant measuring system which generates a signal indicative of a distance between the camera and a target object to be photographed upon receipt of electric power.*	0.1208

Table 8.1: Similarity scores of grammatically correct vs. Incorrect Queries with the first result by PQAI

for the search in the other 30 cases. As a result, the scores we obtained while merely using general statements were poor than when coupled with specific sentences.

In Figure 8.1, the plot demonstrates that the search engine's relevancy was typically substantially higher when both general and specialized sentences were searched simultaneously. That clearly illustrates how crucial it is to include specific information in the query.

8.1.5 Relevancy of Queries with Verbose Content

We compute the features for the queries, similar to how we did it when we were training our length prediction model. We first extract the topic segments from the queries, then extract the features for each segment. We use the length prediction model in chapter 6 to make predictions. One query may occasionally contain multiple topic segments. Our model estimates the length of each topic segment for these queries. On the PQAI engine, the query

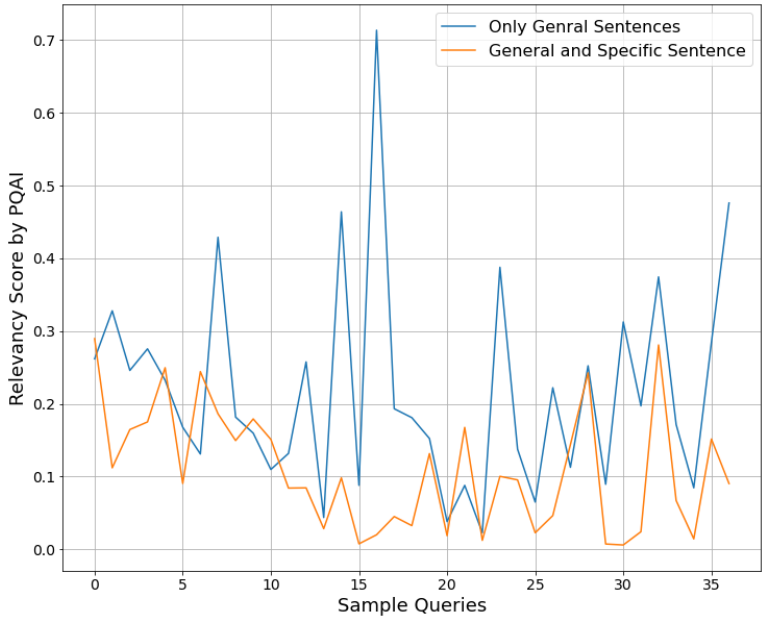

Figure 8.1: Relevacy Scores with PQAI results using queries containing only general sentences vs. queries having general and specific sentences

must be tested in its entirety. In these situations, we take the verbosity degree of the two topic segments and add them.

We used our model to predict the lengths of 91 queries, and the PQAI engine processed those queries. We utilized Quillbot to summarize the queries, which also altered the grammar of the paraphrased queries, to compare the findings with another query form. For these queries, we additionally noted the relevancy score by PQAI.

In **90%** cases where the verbosity degree was lower when compared to their respective paraphrased queries, there was a reduction in the relevancy score. That means queries with lower verbosity degrees performed exceptionally well. However, for 10% cases, we observed poor relevancy scores by PQAI even with a verbose query.

Another pattern we saw was that the query with a negative verbosity degree did well in about 100% of the instances. That indicates that the query was successful even though the predicted length was shorter than the actual length. Table 8.2 and 8.3 show some examples:

Therefore, in most cases, when the verbosity degree is lower, no matter whether it is positive or negative, better scores are achieved. However, the negative verbosity degree score may also indicate that the expected length of the query is lower than its actual length, and it may imply that the query is missing some necessary information that it should contain. A very low predicted length will indicate shortness of text, which may drastically harm the relevance of the search results. Hence it is advised to the user that he/she should not avoid giving all necessary details of their invention.

Queries	Scores
Actual Length = 58, Predicted Length = 79, Verbosity Degree = 21 *A workplace timer app that initiates work and breaks intervals for a person to follow. It reminds users to take appropriate breaks. The user has an option to override the app's suggestions. The app keeps tracks of the degree of user's compliance and analyzes the data to determine the optimum lengths of work and break intervals.*	**0.1497**
Actual Length = 51, Predicted Length = 85, Verbosity Degree = 34 *The app keeps track of the degree of user's compliance and analyzes the data to determine the optimum lengths of work and break intervals. It reminds users to take appropriate breaks. The user has an option to override the app's suggestions for how long a break they should take.*	0.1980

Table 8.2: Two queries but same information. One with lower verbosity degree (top) results in better score

Queries	Scores
Actual Length = 56, Predicted Length = 87, Verbosity Degree = 31 *Fancy yarn is spun using a process that ensures consistency between the produced fancy yarn and its desired configuration. A sensor device in a spinning mechanism continuously measures the diameter of the fancy yarn. Based on the measured diameter values, the fancy configuration of the created yarn is determined and compared to the specified fancy configuration.*	**0.0222**
Actual Length = 87, Predicted Length = 62, Verbosity Degree = -25 *a process that enhances the consistency between the produced fancy yarn and its desired configuration. After being manufactured, the fancy yarn is fed via a sensor device in a spinning mechanism, which continuously measures the diameter of the fancy yarn. Based on the measured diameter values, the fancy configuration of the created yarn is determined and compared to the specified fancy configuration. The comparison is done repeatedly until there is a good match between the predetermined fancy configuration and the fancy configuration of the optimized, generated yarn.*	**0.0082**

Table 8.3: Two queries but the same information. One with a negative verbosity degree (top) results in a better score

9 Conclusion and Future Work

We briefly covered a variety of potential defects in a natural language prior art search query in this thesis, as well as methods to identify them so that they can be avoided and the inventor can obtain relevant prior art for his or her patent. A natural language query may have several problems, but we just focused on three. When a user searches on an AI-based prior art search engine, we offered three models to identify the defects that could lead to the user receiving irrelevant prior art results for their queries.

We also provided an algorithm to add grammatical mistakes to any dataset that contains the text. This approach might be helpful in a variety of grammatical error-related activities for academics. A dataset of numerous grammatical faults, such as wrong verb tenses, article errors, homonym errors, and spelling issues in material from the patent domain, is also included.

In the patent dataset, our specificity model is intelligent enough to differentiate between generic and specific sentences. It would be helpful to the creators of AI based prior art search engines to tell when a user query is missing a crucial piece of information.

The length prediction model did remarkably well at identifying verbose text in our verbosity model. This method can be utilized for numerous text reformation tasks and could be helpful for search engine developers working with existing technology.

Finally, we demonstrated the relevance of the queries against their well-formed counterpart, highlighting the faults we discussed in this thesis. We also demonstrated how certain information affects a prior art query's overall quality.

We anticipate that this thesis can significantly contribute to the field of text analysis and reformation, particularly for texts in the patent domain. However, research in this direction is still in its early stages of development and there are still many tasks that may be accomplished in AI based prior art search engines. We can come up with other flaws that might arise in a natural language query and influence the effectiveness of the search. As we have only worked on finding defects in a patent query, we can also propose solutions to remove these problems from the queries automatically.

Bibliography

[All95] James Allen. *Natural language understanding*. Benjamin-Cummings Publishing Co., Inc., 1995.

[Bro19] Jason Brownlee. *A gentle introduction to transfer learning for Deep learning*. Sept. 2019. URL: https://machinelearningmastery.com/transfer-learning-for-deep-learning/.

[CG16] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[Chu+20] Zewei Chu et al. "How to ask better questions? a large-scale multi-domain dataset for rewriting ill-formed questions". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 7586–7593.

[Dev+18] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[Eli20] Francesco Elia. *Constituency vs dependency parsing*. Oct. 2020. URL: https://www.baeldung.com/cs/constituency-vs-dependency-parsing.

[Fan+08] Rong-En Fan et al. "LIBLINEAR: A library for large linear classification". In: *the Journal of machine Learning research* 9 (2008), pp. 1871–1874.

[FD18] Manaal Faruqui and Dipanjan Das. "Identifying well-formed natural language questions". In: *arXiv preprint arXiv:1808.09419* (2018).

[FFR19] Michael Flor, Michael Fried, and Alla Rozovskaya. "A benchmark corpus of English misspellings and a minimally-supervised model for spelling correction". In: *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*. 2019, pp. 76–86.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[HG14] Clayton Hutto and Eric Gilbert. "Vader: A parsimonious rule-based model for sentiment analysis of social media text". In: *Proceedings of the international AAAI conference on web and social media*. Vol. 8. 1. 2014, pp. 216–225.

[Hut11] Graeme D Hutcheson. "Ordinary least-squares regression". In: *L. Moutinho and GD Hutcheson, The SAGE dictionary of quantitative management research* (2011), pp. 224–228.

[Kim14] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. URL: https://aclanthology.org/D14-1181.

[LN11] Annie Louis and Ani Nenkova. "Automatic identification of general and specific sentences by leveraging discourse annotations". In: *Proceedings of 5th international joint conference on natural language processing*. 2011, pp. 605–613.

[LN14] Annie Louis and Ani Nenkova. "Verbose, laconic or just right: A simple computational model of content appropriateness under length constraints". In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. 2014, pp. 636–644.

[Lou13] Annie Priyadarshini Louis. *Predicting text quality: metrics for content, organization and reader interest*. University of Pennsylvania, 2013.

[Man+14] Christopher D Manning et al. "The Stanford CoreNLP natural language processing toolkit". In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 2014, pp. 55–60.

[Mil95] George A Miller. "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[Mil98] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[Mol20] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[Mwi21] Derrick Mwiti. *Transfer Learning Guide: A Practical Tutorial With Examples for Images and Text in Keras - neptune.ai*. Mar. 2021. URL: `https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras`.

[MZS21] Mahesh Maan, Sam Zellner, and Anirudh Sanutra. "Modular Development in Patent AI Space: A Case Study". In: (2021).

[PE02] Ian Pitt and Alistair Edwards. *Design of speech-based devices: a practical guide*. Springer Science & Business Media, 2002.

[Ped+11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[PN09] Emily Pitler and Ani Nenkova. "Using syntax to disambiguate explicit discourse connectives in text". In: (2009).

[Pra+08] Rashmi Prasad et al. "The Penn Discourse TreeBank 2.0." In: *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. 2008.

[RK19] Julian Risch and Ralf Krestel. "Domain-specific word embeddings for patent classification". In: *Data Technologies and Applications* 53.1 (2019), pp. 108–122.

[Rud+19] Sebastian Ruder et al. "Transfer Learning in Natural Language Processing". In: Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 15–18. DOI: `10.18653/v1/N19-5004`.

[SDS66] Philip J Stone, Dexter C Dunphy, and Marshall S Smith. "The general inquirer: A computer approach to content analysis." In: (1966).

[SLW19] Eva Sharma, Chen Li, and Lu Wang. "Bigpatent: A large-scale dataset for abstractive and coherent summarization". In: *arXiv preprint arXiv:1906.03741* (2019).

[Sol+21] Alessandro Solbiati et al. "Unsupervised topic segmentation of meetings with BERT embeddings". In: *arXiv preprint arXiv:2106.12978* (2021).

[Sor] Mohammad S Sorower. "A literature survey on algorithms for multi-label learning". In: ().

[Sun+19] Chi Sun et al. "How to fine-tune bert for text classification?" In: *China national conference on Chinese computational linguistics*. Springer. 2019, pp. 194–206.

[The+10] Mike Thelwall et al. "Sentiment strength detection in short informal text". In: *Journal of the American society for information science and technology* 61.12 (2010), pp. 2544–2558.

[TL09] José Carlos Toucedo and David E Losada. "Formulating good queries for prior art search". In: *Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer. 2009, pp. 418–425.

[Vas+17] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[Wil90] Joseph Williams. "Toward clarity and grace". In: *Chicago: The University of Chicago* (1990).

[WWH05] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. "Recognizing contextual polarity in phrase-level sentiment analysis". In: *Proceedings of human language technology conference and conference on empirical methods in natural language processing*. 2005, pp. 347–354.

Declaration of Academic Integrity / Eidesstattliche Erklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe und dass alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind. Mit der aktuell geltenden Fassung der Satzung der Universität Passau zur Sicherung guter wissenschaftlicher Praxis und für den Umgang mit wissenschaftlichem Fehlverhalten vom 31. Juli 2008 (vABlUP Seite 283) bin ich vertraut. Ich erkläre mich einverstanden mit einer Überprüfung der Arbeit unter Zuhilfenahme von Dienstleistungen Dritter (z.B. Anti-Plagiatssoftware) zur Gewährleistung der einwandfreien Kennzeichnung übernommener Ausführungen ohne Verletzung geistigen Eigentums an einem von anderen geschaffenen urheberrechtlich geschützten Werk oder von anderen stammenden wesentlichen wissenschaftlichen Erkenntnissen, Hypothesen, Lehren oder Forschungsansätzen.

Passau, 24. März 2023

Deepak Rastogi

I hereby confirm that I have composed this scientific work independently without anybody else's assistance and utilising no sources or resources other than those specified. I certify that any content adopted literally or in substance has been properly identified. I have familiarised myself with the University of Passau's most recent Guidelines for Good Scientific Practice and Scientific Misconduct Ramifications from 31 July 2008 (vABlUP Seite 283). I declare my consent to the use of third-party services (e.g., anti-plagiarism software) for the examination of my work to verify the absence of impermissible representation of adopted content without adequate designation violating the intellectual property rights of others by claiming ownership of somebody else's work, scientific findings, hypotheses, teachings or research approaches.

Passau, 24. März 2023

Deepak Rastogi