UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Towards Preemptive Text Edition using Topic Matching on Corpora

Acácio Filipe Pereira Pinto Correia

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

(2º ciclo de estudos)

Orientador: Prof. Doutor João Paulo Cordeiro
Co-orientador: Prof. Doutor Pedro R. M. Inácio

Covilhã, outubro de 2016

Dedicated to my beloved family.

Acknowledgments

I have received tremendous support throughout my academic life, and this year was no exception. With the help of many people I was able to focus on the work at hand with ease. As such, I would like to thank those who supported me and ensured I was able to finish this dissertation.

A special thanks to my family for always backing me up in my decisions and providing the necessary conditions for me to work. Their love and support has been the most important factor in my personal and professional development.

I am very grateful for the help and guidance that was given to me by both my supervisors, without whom I would be unable to deliver this work. I would like to thank Professor João Paulo Cordeiro and Professor Pedro Ricardo Morais Inácio for their wisdom and advisement.

I acknowledge that meeting with my friends has always been a good stress reliever, providing me with lots of joy and fun. Thank you for all the moments that enriched my day. I am grateful to João Neves for the interesting discussions that allowed me to develop a better critical thinking.

Finally, I would like to thank Musa Samaila, Manuel Meruje and specially Bernardo Sequeiros, mostly for the great environment they have created in the laboratory and their help in the development of this work.

Resumo

Hoje em dia, a realização de uma investigação científica só é valorizada quando resulta na publicação de artigos científicos em jornais ou revistas internacionais de renome na respetiva área do conhecimento. Esta perspetiva reflete a importância de que os estudos realizados sejam validados por pares. A validação implica uma análise detalhada do estudo realizado, incluindo a qualidade da escrita e a existência de novidades, entre outros detalhes. Por estas razões, com a publicação do documento, outros investigadores têm uma garantia de qualidade do estudo realizado e podem, por isso, utilizar o conhecimento gerado para o seu próprio trabalho. A publicação destes documentos cria um ciclo de troca de informação que é responsável por acelerar o processo de desenvolvimento de novas técnicas, teorias e tecnologias, resultando na produção de valor acrescido para a sociedade em geral.

Apesar de todas estas vantagens, a existência de uma verificação detalhada do conteúdo do documento enviado para publicação requer esforço e trabalho acrescentado para os autores. Estes devem assegurar-se da qualidade do manuscrito, visto que o envio de um documento defeituoso transmite uma imagem pouco profissional dos autores, podendo mesmo resultar na rejeição da sua publicação nessa revista ou ata de conferência. O objetivo deste trabalho é desenvolver um algoritmo para ajudar os autores na escrita deste tipo de documentos, propondo sugestões para melhoramentos tendo em conta o seu contexto específico.

A ideia genérica para solucionar o problema passa pela extração do tema do documento a ser escrito, criando sugestões através da comparação do seu conteúdo com o de documentos científicos antes publicados na mesma área. Tendo em conta esta ideia e o contexto previamente apresentado, foi realizado um estudo de técnicas associadas à área de Processamento de Linguagem Natural (PLN). O PLN fornece ferramentas para a criação de modelos capazes de representar o documento e os temas que lhe estão associados. Os principais conceitos incluem n-grams e modelação de tópicos (*topic modeling*). Para concluir o estudo, foram analisados trabalhos realizados na área dos artigos científicos, estudando a sua estrutura e principais conteúdos, sendo ainda abordadas algumas características comuns a artigos de qualidade e ferramentas desenvolvidas para ajudar na sua escrita.

O algoritmo desenvolvido é formado pela junção de um conjunto de ferramentas e por uma coleção de documentos, bem como pela lógica que liga todos os componentes, implementada durante este trabalho de mestrado. Esta coleção de documentos é constituída por artigos completos de algumas áreas, incluindo Informática, Física e Matemática, entre outras. Antes da análise de documentos, foi feita a extração de tópicos da coleção utilizada. Deste forma, ao extrair os tópicos do documento sob análise, é possível selecionar os documentos da coleção mais

semelhantes, sendo estes utilizados para a criação de sugestões. Através de um conjunto de ferramentas para análise sintática, pesquisa de sinónimos e realização morfológica, o algoritmo é capaz de criar sugestões de substituições de palavras que são mais comummente utilizadas na área.

Os testes realizados permitiram demonstrar que, em alguns casos, o algoritmo é capaz de fornecer sugestões úteis de forma a aproximar os termos utilizados no documento com os termos mais utilizados no estado de arte de uma determinada área científica. Isto constitui uma evidência de que a utilização do algoritmo desenvolvido pode melhorar a qualidade da escrita de documentos científicos, visto que estes tendem a aproximar-se daqueles já publicados. Apesar dos resultados apresentados não refletirem uma grande melhoria no documento, estes deverão ser considerados uma baixa estimativa ao valor real do algoritmo. Isto é justificado pela presença de inúmeros erros resultantes da conversão dos documentos pdf para texto, estando estes presentes tanto na coleção de documentos, como nos testes.

As principais contribuições deste trabalho incluem a partilha do estudo realizado, o desenho e implementação do algoritmo e o editor de texto desenvolvido como prova de conceito. A análise de especificidade de um contexto, que advém dos testes realizados às várias áreas do conhecimento, e a extensa coleção de documentos, totalmente compilada durante este mestrado, são também contribuições do trabalho.

Palavras-chave

Artigos, Documentos Científicos, Language Tool, Latent Dirichlet Allocation, LDA, N-gram, Processamento de Linguagem Natural, PLN, Qualidade, Sugestões, Wordnet.

Resumo alargado

Introdução

Este capítulo serve de resumo ao trabalho descrito nesta dissertação, expandindo um pouco mais o que foi exposto no resumo. A primeira subsecção apresenta o enquadramento da dissertação, o problema que se pretende analisar e os objetivos propostos para este trabalho. Depois seguem-se as principais contribuições, o estado da arte e o desenho e implementação do algoritmo desenvolvido no contexto do projeto. Finalmente, as últimas subsecções apresentam os testes mais importantes e as principais conclusões extraídas do trabalho realizado.

Enquadramento, Descrição do Problema e Objetivos

As sociedades evoluem através da criação, recolha e utilização do conhecimento. A sua utilização permite o desenvolvimento de ideias que podem resultar em melhorias na qualidade de vida. Uma das principais fontes de conhecimento é o trabalho realizado pela comunidade científica. Este consiste fundamentalmente na experimentação, com o objetivo de testar uma ideia, provar um resultado teórico ou procurar novas soluções para um problema existente, entre outros. Quando estas experiências apresentam resultados promissores, é escrito um artigo científico que descreve a experiência realizada, as ideias que motivaram este desenvolvimento e procedimentos seguidos para a sua realização. Estes artigos são depois submetidos a revistas e jornais científicos da área, que são responsáveis por assegurar a qualidade do documento, garantindo a existência de novidade na experiência e verificando a qualidade da escrita. Quando aceites, os artigos publicados resultam na partilha de conhecimento dentro da comunidade científica, possibilitando o posterior desenvolvimento desse estudo por outros investigadores, ou servindo de inspiração para a realização de outros estudos. Este processo de publicação resulta num ciclo de partilha de informação que é responsável por acelerar o progresso do conhecimento e assegurar a correção do trabalho desenvolvido.

O facto de ser realizada uma verificação do trabalho apresentado implica um esforço acrescido para o investigador, visto que este deve assegurar a qualidade do documento. A falta de qualidade de um documento pode transmitir uma imagem pouco profissional dos seus autores, dificultando a sua publicação numa comunidade onde esta é a única forma de valorizar a investigação desenvolvida [CK11]. A escrita deste tipo de documentos é uma prática difícil [RKEO15], requerendo um elevado grau de conhecimento da linguagem utilizada. Assegurar que o texto escrito transmite corretamente a mensagem desejada pelo autor é uma das dificuldades sentidas pelos investigadores, visto que o significado de um termo depende do contexto no qual este está inserido. A utilização de uma terminologia específica para cada área do conhecimento é outra dificuldade encontrada.

Dado o problema apresentado, os principais objetivos deste trabalho são:

- O desenvolvimento de um algoritmo capaz de assistir um investigador na escrita de documentos científicos, onde o contexto do documento é tido em conta. Este desenvolvimento pode ser repartido em dois sub-algoritmos: o primeiro é responsável por extrair o contexto do documento e encontrar fontes de informação de acordo com esse contexto; e o segundo é responsável pelo cálculo de sugestões de melhorias e de correções de acordo com a informação recolhida das fontes selecionadas pelo primeiro;
- A avaliação do desempenho do algoritmo desenvolvido. Esta deve ser acompanhada da configuração do algoritmo de forma a maximizar o seu desempenho, de acordo com os resultados obtidos;
- A implementação de um editor de texto com a integração do algoritmo. Este último objetivo serve como prova de conceito da utilização do algoritmo num contexto realista.

Principais Contribuições

As principais contribuições resultantes do trabalho realizado no âmbito deste projeto podem ser sumariamente descritas da seguinte forma:

- A primeira contribuição é a apresentação de um estudo ao estado da arte de conceitos relacionados com o trabalho desenvolvido. Este estudo descreve alguns conceitos introdutórios relativamente à área de Processamento de Linguagem Natural (PLN), incluindo modelação de linguagem através de n-grams e algumas técnicas de *smoothing*, utilizadas na resolução do problema da escassez de dados (*data sparsity*). A segunda parte do estudo apresenta algoritmos de modelação de tópicos (*topic modeling*) para extração de temas de uma coleção de documentos. O estudo contém ainda a apresentação de trabalhos realizados na área da escrita e avaliação de documentos científicos;
- O algoritmo capaz de propor sugestões relativamente a melhorias ou correções, tendo em conta o contexto específico do documento que se encontra sobre análise é outra contribuição importante. O código fonte do editor de texto que instancia este algoritmo irá ser aberto, permitindo o estudo e desenvolvimento de melhorias pela comunidade;
- A aplicação desenvolvida como prova de conceito do algoritmo é ela própria uma contribuição visto que permitirá aos investigadores receber sugestões para os seus documentos;
- A análise dos resultados de um sistema que combina este conjunto específico de ferramentas e técnicas de PLN não foi encontrada no estado da arte, sendo também esta uma contribuição;
- Depois de desenvolvido o algoritmo foram realizados testes para determinar o desempenho do mesmo. Os testes realizados utilizaram um conjunto de tabelas com n-grams de vários

géneros de Inglês. Este conjunto inclui: tabelas com n-grams de Inglês genérico; e várias tabelas com n-grams específicos de cada uma das áreas contidas na coleção de documentos utilizada. A variedade de contextos descrita pelas tabelas utilizadas permitiu a realização de um pequeno estudo relativo à especificidade da escrita em documentos científicos.

A coleção de documentos recolhida é outra contribuição que resultou como colateral do trabalho realizado. Após uma extensa procura, sem sucesso, por coleções de documentos, que apresentassem as características necessárias à realização deste trabalho, a solução encontrada foi recolher documentos para a construção de uma nova coleção. Esta coleção segue uma aproximação à distribuição uniforme relativamente ao número de documentos de cada área (e subárea), contendo mais de trinta mil documentos das áreas de Informática, Matemática, Física, Estatística, Biologia Quantitativa e Finanças Quantitativas.

Estado da Arte

Os n-grams são o modelo mais utilizado para a representação do texto de um subconjunto de uma linguagem [CG96]. Como tal, este é o principal foco da secção inicial do Capítulo 2, onde é descrito o processo utilizado para o cálculo da probabilidade de expressões ou frases. Segue-se uma descrição do problema, que se depara com o facto da maioria dos n-grams nunca ocorrer no conjunto de treino (*data sparsity*), bem como da sua resolução através de técnicas de *smoothing*, que permitem atribuir uma probabilidade a estes n-grams. São apresentadas algumas técnicas de *smoothing*, culminando na versão modificada do *Kneser-Ney* que é aquela que apresenta melhor desempenho [CG96]. Para concluir esta secção são apresentados os métodos utilizados na avaliação deste tipo de técnicas.

A secção do Capítulo 2 que se segue é referente à modelação de tópicos (*topic modeling*) utilizada na extração de temas de um conjunto de documentos. O resultado da modelação de tópicos é a representação reduzida de cada um dos documentos na coleção analisada de acordo com os tópicos extraídos. Os algoritmos apresentados nesta secção são classificados como algoritmos de *generative probabilistic topic modeling*, visto que assentam na ideia de que cada documento na coleção foi gerado através de um processo chamado generativo (*generative*). O objetivo destes algoritmos é o de reconstruir a estrutura associada ao processo generativo, onde estão definidas as representações de cada um dos documentos, entre outras variáveis. São então apresentados alguns dos métodos utilizados para estimação desta estrutura. Esta secção finaliza com a descrição de procedimentos que podem ser utilizados para comparar documentos através da representação obtida por este processo.

Para terminar o Capítulo 2, o estudo do estado da arte apresenta um conjunto de trabalhos de investigação realizados na área dos documentos científicos. Esta secção começa por apresentar estudos que descrevem uma das estruturas mais utilizadas neste tipo de documentos, a estrutura Introduction Methods Results and Discussion (IMRAD). Para esta estrutura, são apresentadas

algumas teorias sobre o intuito dos segmentos apresentados em cada uma das secções. A secção que se segue apresenta algumas características relacionadas com o estudo da qualidade de um documento científico e uma ferramenta para análise automática de ensaios. Finalmente, é descrito um conjunto de ferramentas desenvolvido com um intuito similar ao deste trabalho, que é o de ajudar na escrita de documentos científicos.

Algoritmo

O algoritmo desenvolvido pode ser dividido em dois sub-algoritmos: o primeiro é responsável por extrair os temas tratados no documento, de forma a conseguir selecionar um conjunto de documentos cujos temas sejam semelhantes; e o segundo é responsável por calcular sugestões para melhoramentos e correções de palavras num documento, de acordo com os documentos selecionados pelo primeiro sub-algoritmo. Este primeiro sub-algoritmo baseia-se principalmente na utilização de modelação de tópicos (*topic modeling*) para encontrar documentos semelhantes num corpus. O segundo recorre a um conjunto de técnicas para criar uma lista de candidatos a possíveis substituições de palavras existentes no documento. Depois, através da utilização de um modelo de n-grams pré calculado nos documentos selecionados, o algoritmo propõe ao utilizador os três candidatos mais prováveis de aparecer.

Testes

Os testes realizados tinham como principal objetivo avaliar o desempenho do algoritmo desenvolvido. Foram realizados testes de dois tipos: objetivos, onde foram avaliadas as sugestões propostas pelo algoritmo em documentos com alterações realizadas automaticamente; e subjetivos, através da avaliação subjetiva de várias versões de um conjunto de parágrafos, em que alguns aplicavam as sugestões propostas pelo algoritmo.

Os testes objetivos verificavam se o algoritmo era capaz de propor como sugestões as palavras que estavam no documento antes da sua alteração. A avaliação foi realizada de acordo com o Mean Reciprocal Rank (MRR), permitindo uma análise da correção das palavras tendo em conta a posição em que a palavra correta ocorre, valorizando pouco palavras que aparecem nas posições mais baixas da lista. O melhor valor de MRR obtido diz respeito à utilização dos n-grams de todos os documentos do conjunto de treino de Informática, como fonte para a análise dos documentos do conjunto de teste de Informática.

Os testes subjetivos analisaram a opinião de um pequeno conjunto de sujeitos em relação a várias versões de um mesmo conjunto de parágrafos. O primeiro conjunto tinha sido escrito pelo autor deste documento sem recorrer a quaisquer análises externas. O segundo resultou da aplicação da melhor sugestão proposta pelo algoritmo para todas as palavras cuja própria palavra não estivesse na lista de sugestões, para o conjunto original. O terceiro resulta de um processo semelhante ao segundo com um maior grau de liberdade, onde a aplicação de sugestões

só era feita nos melhores casos, permitindo ainda a conjugação das sugestões propostas. Os resultados destes testes demonstraram que a maioria dos sujeitos concordou que os parágrafos provenientes deste último processo eram aqueles cujo vocabulário era o melhor.

Conclusões e Trabalho Futuro

Os objetivos foram atingidos, sendo o principal o desenvolvimento de um algoritmo com o objetivo de ajudar na escrita de artigos científicos e os restantes a sua avaliação e posterior integração num editor de texto.

Os resultados obtidos através dos testes objetivos apresentam uma clara aproximação do documento analisado aos documentos utilizados para o cálculo das sugestões. No melhor caso, os testes demonstraram que o algoritmo era capaz de recuperar mais de um terço das palavras originais que foram alteradas automaticamente. Isto é uma indicação de que algumas características linguísticas estão a ser capturadas pelo algoritmo.

Os testes subjetivos apresentaram resultados menos claros, onde, ainda que por pouco, a maioria dos sujeitos selecionou mais vezes os parágrafos sem nenhuma das alterações fornecidas pelo algoritmo. Uma possível justificação depara-se com o facto da lista de sugestões utilizada conter todos os candidatos, sem uma seleção prévia dos três mais prováveis. Isto pode ter induzido em alterações onde a palavra substituta era menos provável do que aquela já apresentada nos parágrafos originais.

A combinação dos resultados obtidos em ambos os tipos de teste parecem indicar que a utilização do algoritmo pode ajudar a ultrapassar algumas das dificuldades sentidas pelos investigadores aquando da escrita de documentos científicos. No entanto, esta melhoria está dependente da realização de uma análise detalhada das sugestões propostas, tornado este processo moroso e demorado. Como tal, o melhor será talvez recorrer às sugestões apenas em palavras onde o autor tenha dúvidas, diminuindo o esforço necessário mas continuando a beneficiar do seu uso.

Abstract

Nowadays, the results of scientific research are only recognized when published in papers for international journals or magazines of the respective area of knowledge. This perspective reflects the importance of having the work reviewed by peers. The revision encompasses a thorough analysis on the work performed, including quality of writing and whether the study advances the state-of-the-art, among other details. For these reasons, with the publishing of the document, other researchers have an assurance of the high quality of the study presented and can, therefore, make direct usage of the findings in their own work. The publishing of documents creates a cycle of information exchange responsible for speeding up the progress behind the development of new techniques, theories and technologies, resulting in added value for the entire society.

Nonetheless, the existence of a detailed revision of the content sent for publication requires additional effort and dedication from its authors. They must make sure that the manuscript is of high quality, since sending a document with mistakes conveys an unprofessional image of the authors, which may result in the rejection at the journal or magazine. The objective of this work is to develop an algorithm capable of assisting in the writing of this type of documents, by proposing suggestions of possible improvements or corrections according to its specific context.

The general idea for the solution proposed is for the algorithm to calculate suggestions of improvements by comparing the content of the document being written in to that of similar published documents on the field. In this context, a study on Natural Language Processing (NLP) techniques used in the creation of models for representing the document and its subjects was performed. NLP provides the tools for creating models to represent the documents and identify their topics. The main concepts include n-grams and topic modeling. The study included also an analysis of some works performed in the field of academic writing. The structure and contents of this type of documents, the presentation of some of the characteristics that are common to high quality articles, as well as the tools developed with the objective of helping in its writing were also subject of analysis.

The developed algorithm derives from the combination of several tools backed up by a collection of documents, as well as the logic connecting all components, implemented in the scope of this Master's. The collection of documents is constituted by full text of articles from different areas, including Computer Science, Physics and Mathematics, among others. The topics of these documents were extracted and stored in order to be fed to the algorithm. By comparing the topics extracted from the document under analysis with those from the documents in the collection, it is possible to select its closest documents, using them for the creation of suggestions. The

algorithm is capable of proposing suggestions for word replacements which are more commonly utilized in a given field of knowledge through a set of tools used in syntactic analysis, synonyms search and morphological realization.

Both objective and subjective tests were conducted on the algorithm. They demonstrate that, in some cases, the algorithm proposes suggestions which approximate the terms used in the document to the most utilized terms in the state-of-the-art of a defined scientific field. This points towards the idea that the usage of the algorithm should improve the quality of the documents, as they become more similar to the ones already published. Even though the improvements to the documents are minimal, they should be understood as a lower bound for the real utility of the algorithm. This statement is partially justified by the existence of several parsing errors both in the training and test sets, resulting from the parsing of the pdf files from the original articles, which can be improved in a production system.

The main contributions of this work include the presentation of the study performed on the state of the art, the design and implementation of the algorithm and the text editor developed as a proof of concept. The analysis on the specificity of the context, which results from the tests performed on different areas of knowledge, and the large collection of documents, gathered during this Master's program, are also important contributions of this work.

Keywords

Language Tool, Latent Dirichlet Allocation, LDA, Natural Language Processing, NLP, N-gram, Papers, Quality, Scientific Documents, Suggestions, Wordnet.

Contents

List of Figures

List of Tables

Acronyms

ACM Association for Computing Machinery

AES Automatic Essay Scoring

API Application Programming Interface

CARS Create A Research Space

CCS Computing Classification System

COCA Corpus of Contemporary American English

DBMS Database Management System

DP Dirichlet Process

EM Expectation Maximization

HDP Hierarchical Dirichlet Processes

HTML5 HyperText Markup Language 5

IMRAD Introduction Methods Results and Discussion

JWI Java Wordnet Interface

KL Kullback Leibler

LDA Latent Dirichlet Allocation

MALLET MAchine Learning for LanguagE Toolkit

MCMC Monte Carlo Markov Chain

MIT Massachusetts Institute of Technology

ML Maximum Likelihood

MRR Mean Reciprocal Rank

NLP Natural Language Processing

OS Operating System

PEG Project Essay Grade

PLSI Probabilistic Latent Semantic Indexing

POS Part of Speech

TEM Tempered Expectation Maximization

TNG Topical N-Gram

UBI Universidade da Beira Interior

XIP Xerox Incremental Parser

Nomenclature

DT Determiner

IN Preposition or subordinating conjunction

JJ Adjective

NN Noun, singular or mass

NNS Noun, plural

NNP Proper noun, singular

VBG Verb, gerund or present participle

VBN Verb, past participle

VBP Verb, non-3rd person singular present

VBZ Verb, 3rd person singular present

WDT Wh-determiner

Chapter 1

Introduction

This document concerns the work performed under the scope of the project for attaining a Master's degree in Computer Science and Engineering at the Universidade da Beira Interior (UBI). The main subject of the dissertation is the study and development of an algorithm to assist in the writing of scientific documents. As a proof of concept, the implementation of the algorithm was included in a very simple text editing tool, analyzed later in the document. The subsequent sections of the chapter describe the scope of this dissertation, the motivation that led to the realization of the project, the associated problem and the fundamental objectives. Afterwards, the chapter includes a description of the adopted approach, main contributions and a section with the organization of the remaining parts of the document.

1.1 Motivation and Scope

Societies evolve through the creation, gathering and use of knowledge. This knowledge can then be levered to create tools and develop ideas that sometimes improve quality of life. The main point of origin for this knowledge is the scientific community and its experiments, which are performed with the goal of testing an idea, proving a theoretical result, searching for new algorithms, tools or novel adaptations for using old tools, among others. When an experiment presents promissory results, a scientific paper describing the main findings, ideas and procedures that led to such development is written and submitted to scrutiny. By submitting these documents to scientific journals and magazines of the area, the work gets analyzed by scientific committees that verify the novelty, scientific correctness, and quality of writing of the proposed text. If accepted, the publication results in the sharing of the experiment with the remaining scientific community, allowing for other investigators to work on those findings and improving that work, or taking inspiration for new developments in that or different areas. This process results in a cycle of information sharing responsible for speeding up the progress of science and ensuring the correctness of the work performed.

"Research writing is classified as a type of academic writing. Therefore, it is considered formal writing." [NSH12]. Learning to write in this context is a challenging task [RKEO15], since there are many important factors that should be thoroughly considered, including the selection of appropriate vocabulary, correct use of grammar and following a strict writing structure. These factors and the writing of persuasive arguments, necessary for effectively conveying information, are specially hard for beginners [UU15, LS15] and non-native English re-

searchers [NSH12, CK11, LS15].

Since the time humans realized the true potential of computers to these days, we have been restricted to a set of predefined interactions with machines. These interactions are usually performed through the selection of options or done by the introduction of very specific instructions. Deviating from these interactions results in failing to perform the desired actions. The objective of Natural Language Processing is to retrieve information from a natural language input. In theory, this could allow a user to introduce text or speak to a machine, without restrictions, and the machine would be capable of identifying the intention of the user and responding accordingly. This scenario would provide users with major improvements both in terms of comfort as well as effectiveness in terms of response. These serve as motivation for abundant researches and studies in this area, each tackling a different task, including analysis on the semantic and extraction of the syntactic structure of an input. However, due to the complexity inherent to the study of natural languages, the creation of a model capable of capturing a natural language in its entirety is still an utopia, continuing limited to smaller tasks. Polisemy, which is the possibility of words having more than one meaning, depending on the context, and ambiguity of interpretation are just some examples of problems that investigators have to face in this field of expertise, justifying the limitations in progress made so far.

In this context and while fully understanding natural language is still just an idea, suggestions have been introduced as a way of improving humans interaction both in speed and efficiency, as in the cases of word completion and prediction available in most mobile devices [vdBB08]. Their utilization can also impact effectiveness, by providing users with what they most likely need or want, as is the case of search engines. These advantages have molded the human mind into accepting suggestions as a beneficial tool for their interactions, paving the way for new tools and technologies that implement this sort of techniques.

The aforementioned reasons are on the basis of the proposal for the development of an algorithm designed to provide the user with suggestions for improvements in academic texts. The suggestions should include corrections for grammatical errors, such as orthographic errors, syntax and semantics, already provided by some tools. Nonetheless, the main focus of this work concerns the misuse of terms in a specific field of knowledge. This algorithm, and its inherent study, should improve the understanding on the specificity of the syntactic structures in academic writing.

The area that best describes the scope of this Computer Science and Engineering Master's project is natural language processing, as it focuses on presenting suggestions for possible improvements to the text being written in the academic field. Under the 2012 version of the ACM Computing Classification System (CCS), the de facto standard for Computer Science, the scope of this dissertation can be described by the following topics:

- **Computing methodologies~Natural language processing**;
- *Information systems~Language models*;
- *Information systems~Document topic models*;
- Applied computing~Text editing.

1.2 Problem Statement and Objectives

The problem addressed in this dissertation is closely related with the difficulties that investigators have to face when writing scientific documents. A major difficulty is ensuring that a sentence conveys the exact meaning the author intends to. Since natural languages are intrinsically ambiguous and imprecise, words can have multiple interpretations depending on the context. Correctly selecting the appropriate words requires experience and knowledge on the language being used. This knowledge is something most non-native English speakers lack, making them more susceptible to incorrectly select words which have a meaning that, in the given context, is different from the one intended.

The terminology is yet another difficulty faced when entering a new area of expertise. Each area makes use of specialized terms and expressions, which are used in detriment of others with a similar meaning. These words and expressions evolved alongside the area, and now provide a very specific meaning, which is difficult to express in a different way. Failing to use them creates inconsistent manuscripts, hardly publishable or accepted by other investigators, since they transmit the image of an unprofessional investigation.

Given the problem statement, the main objectives proposed for this dissertation are:

1. The development of an algorithm for assisting the writing of scientific documents. The algorithm can be divided into two sub-algorithms: one should be capable of identifying the context of a document and searching for related sources; and the second should create suggestions for term replacement and correction of text, according to the context recovered by the first one;
2. The next objective is the evaluation of the performance and fine tuning of the algorithm;
3. The final objective serves as a proof of concept, which is the development of a simple text editing tool with the integration of the aforementioned algorithm.

1.3 Adopted Approach for Solving the Problem

The approach taken to solve this problem and meeting the objectives, included the following steps:

1. The first step was to study the state of the art, including NLP concepts, techniques and technologies. This was followed by a study into the specific area of academic writing, analyzing the work of other investigators and gathering the knowledge which could be levered for the current research;
2. The study mentioned in the previous step, allowed for a better understanding on which techniques and technologies could be used for each of the desired purposes. With this knowledge, the design of the algorithm became the next step;
3. Both the language modeling and the topics modeling, used in the extraction of the context of a document, required a corpus of scientific documents. To fulfill this need the next step became the search for a corpus with the desired characteristics;
4. After a thorough search without obtaining a viable corpus, an alternative procedure consisting of gathering enough documents for the creation of a corpus that could fit the specific needs of this work was pursued;
5. With the corpus complete, the next step was to study the extraction of n-grams from the collection of scientific papers, and, subsequently, to calculate and store the extracted n-grams;
6. The exploration of tools that implemented the techniques and technologies defined in the design of the algorithm formed the next steps;
7. The remaining step for completing the implementation of the algorithm was the combination of all the selected tools;
8. The eighth step was the testing and, after analyzing the results, fine-tunning of the algorithm;
9. The next to last step consisted on the software engineering process for the implementation of a text editor that would integrate the developed algorithm;
10. The final step was the implementation of the text editor with the inclusion of simple functions common to most text editors.

1.4 Main Contributions

The work and research performed within the context of this project resulted in a set of contributions for the advance of scientific knowledge, which can be summarized as follows:

- The first contribution is the presentation of a brief study on the related work. This study features general concepts related to NLP theory and techniques, including a discussion about n-grams and smoothing techniques. A second section presents topic modeling algorithms used in the extraction of topics from a collection of documents. Given the nature

of the dissertation the study then shifts to the specific realm of academic writing, with a more refined development;

- The sub-algorithm developed for the identification of a context and finding related sources and the sub-algorithm that calculates suggestions, according to the defined context, are another important contribution. The complete source code of the text editing tool will be open source, allowing for the study and improvement of the algorithm by the community;
- The delivery of an application capable of assisting the users in the writing of scientific documents with some success, integrating the aforementioned algorithm, constitutes a contribution as well;
- The combination and analysis of the results provided by a system formed with several known techniques and tools for NLP is not described in the literature, to the best of the knowledge of the author;
- After the development of the algorithm, test suites were created with the purpose of refining the results. The tests suites included tests on a variety of tables, with n-grams extracted from different subsets of English. The aforementioned tables comprised: a table with one million of the most frequent trigrams from generic English, from the Corpus of Contemporary American English (COCA); and one set of tables for each of the contemplated disciplines, with the n-grams from the respective documents. The variety of contexts presented in the tables allowed studying the importance of the specificity of the context on academic writing.

The corpus of scientific documents is another contribution that resulted as a bi-product of this work. During the planning phase it became clear that the extraction of context from a document was necessary for providing appropriate suggestions. The use of topic modeling for this purpose required a corpus of scientific documents for the creation of topics. An extensive search for existing corpus proved unfruitful when the only corpus found, with the desired characteristics, SciTex [DOKLK+13], was unavailable for download. The solution followed was to create a corpus by gathering documents from the electronic archive, arXiv [Lib98]. The corpus follows an approximation to an uniform distribution over the areas contemplated, including over thirty thousand scientific documents from Computer Science, Mathematics, Physics, Statistics, Quantitative Biology and Quantitative Finance.

1.5 Dissertation Organization

The dissertation is divided into five main chapters and two appendices. Their contents can be briefly described as follows:

- Chapter 1 — `Introduction` — describes the context and motivation that led to the devel-

opment of this work, the problem to be addressed and the proposed objectives. These sections are followed by the adopted approach for solving the problem, the main contributions to the advance of knowledge and the document organization;

- Chapter 2 — `State of the art` — presents concepts and techniques related to NLP. It then focuses on presenting some of the work done in the specific area of academic writing. A comparative analysis on similar tools and their functionalities is also included in this chapter;
- Chapter 3 — `Design and Implementation` — presents the developed algorithm, the components involved and how they interact with each other. The corpus is yet another subject contained within this chapter;
- Chapter 4 — `Tests and Prototype` — contains the specification of the tests performed to evaluate the algorithm and the discussion of the respective results. These are divided into objective and subjective test sets. The remaining parts of the chapter introduce the text editing tool and describes its functioning;
- Chapter 5 — `Conclusions and Future Work` — discusses the results, analyses the objectives that were accomplished, and presents some of the reasons that justify those that were not met. The chapter then concludes with the description of a small set of features and improvements that could be used to extend and improve this work;
- Appendix A — `Software Engineering` — describes the process of software engineering followed in the development of the text editing tool, used as a proof of concept for the developed algorithm. The appendix includes the requirement specification, use cases and both activity and class diagrams;
- Appendix B — `Results` — presents the results for all the tests performed on the algorithm, and the resulting sections used in the subjective testing.

Chapter 2

State of the Art

2.1 Introduction

This chapter presents the main methods and techniques related with this work. Helping authors achieve a high standard quality in terms of the linguistic constructions used was the primary concern and motivation for this work. Therefore, a study on NLP related issues, such as *language modeling* and *topic modeling*, is presented along with its corresponding importance duly justified. The chapter starts with a general study on NLP, namely language modeling (Section 2.2). It then proceeds with the description of techniques used in topic modeling (Section 2.3) and with a study focused on scientific text standards (Section 2.4).

2.2 General Language Modeling

Language Modeling concerns mainly the knowledge and usage of language patterns in human language, which is a dynamic phenomenon, constantly evolving through time [Vog00]. For instance, the *Portuguese* language used by Luís de Camões, or the *English* language from William Shakespeare, are substantially different from their contemporary versions. Language is a communication protocol grounded by social convention and community agreement. Language evolves over time. Even in a given time period, different language patterns are employed in different text genres, as one can easily recognize by comparing texts from soap operas with scientific texts. Even within the scientific domain, there are variations where different stylistic features and sentence patterns are more likely employed in certain areas than others. This work is especially focused on the study and use of the linguistic features characterizing scientific text production, in order to assist an author in his or her work. The information presented in this section is primarily based on [CG96, JM00], following a similar structure and presentation.

2.2.1 N-grams

In general, language modeling is grounded on discovering the probability of a sequence of words in a predefined context. Considering a sentence s constituted of l words ($w_1 w_2 \dots w_l$), by following the chain rule of probability, one could calculate the joint probability with:

$$\mathbb{P}(s) = \mathbb{P}(w_1)\mathbb{P}(w_2|w_1)\mathbb{P}(w_3|w_1 w_2)\dots\mathbb{P}(w_l|w_1\dots w_{l-1}) = \prod_{i=1}^{l}\mathbb{P}(w_i|w_1\dots w_{i-1}), \quad \textbf{(2.1)}$$

which represents the $\mathbb{P}(s)$ as the product of the conditional probability of each word given every previous word.

Due to the practical difficulties in calculating the probability of every sequence necessary in the computation of larger sequences of text using Equation 2.1, a simplification was created based on the Markov Assumption [JM00]. The Markov Assumption determines that the probability of a future event (word) can be predicted by its nearest past, instead of its full past [JM00]. With this assumption, one may calculate the probability of a bigger sequence resorting only to the $n-1$ words previous to each word, instead of using all the previous words. These sequences of n words (w_i plus the $n-1$ previous words) used in the calculation of the probability of bigger sequences are called *n-grams*, presenting the most widely-used language models [CG96] (where n is the order of the model). The adaptation of Equation 2.1 with the use of n-grams is:

$$\mathbb{P}(s) \approx \prod_{i=n}^{l} \mathbb{P}(w_i | w_{i-n+1} \dots w_{i-1}). \quad (2.2)$$

In Equation 2.2, i starts at n because n-grams are being used and, as such, an n-gram needs $n-1$ previous words, which do not exist before word n. To calculate the probability of a sentence starting with $i = 1$, one could consider the existence of $n-1$ special words preceding the sentence. For similar reasons one may also consider adding special words after the sentence. In this context, word sequences of the type $w_1 w_2 \dots w_l$ are usually represented as w_1^l.

Given that with each operation the probability tends to reduce, using the Equation 2.2 in real situations would easily result in underflow. Considering the computational limitations, the probability of an expression is usually calculated by the sum of the logarithm of the probability of each of its n-grams as in:

$$\log(\mathbb{P}(s)) \approx \sum_{i=1}^{l} \log(\mathbb{P}(w_i | w_{i-n+1}^{i-1})). \quad (2.3)$$

With Equation 2.3, one can compute the probability of large sequences of text based on the probability of n-grams, but notice that the definition of how to compute the probability of an n-gram was not yet presented. The probability of a given n-gram can be estimated by dividing the count of that n-gram in a collection (corpus) of texts by the count of its prefixed (n-1)-gram, previously computed [JM00], as in:

$$\mathbb{P}(w_i | w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})}, \quad (2.4)$$

where $c(w_{i-n+1}^{i-1})$ is the count of the times the previous words occur in the corpus. This value is more commonly represented by $\sum_{w_i} c(w_{i-n+1}^{i})$, in the sense that it can be the sum of the counts for all n-grams where the previous words are w_{i-n+1}^{i-1}, with the advantage that this will

work even for unigrams. This form of estimation is called the Maximum Likelihood (ML), because it provides the highest possible probability for the data that appeared on the corpus (the training data) [CG96].

One is now capable of computing the probability of any sequence of text. As a toy example[1], consider that the text in Example 1 is the corpus and that the objective is to calculate the probability of the second sentence (`They live in New York.`) using trigrams (n-grams where $n = 3$).

> Example 1. We live in San Francisco, more precisely near San Francisco bay.
> They live in New York.
> Their parents live outside, in the middle of the jungle.

Using Equation 2.2 the probability of the sentence can be written as:

$$\mathbb{P}(s) = \mathbb{P}(\text{"}in\text{"}|\text{"}They\text{" "}live\text{"})\mathbb{P}(\text{"}New\text{"}|\text{"}live\text{" "}in\text{"})\mathbb{P}(\text{"}York\text{"}|\text{"}in\text{" "}New\text{"}),$$

whose individual values can be obtained resorting to Equation 2.4, as follows:

$$\mathbb{P}(\text{"}in\text{"}|\text{"}They\text{" "}live\text{"}) = \frac{c(\text{"}They\ live\ in\text{"})}{c(\text{"}They\ live\text{"})} = \frac{1}{1} = 1,$$

$$\mathbb{P}(\text{"}New\text{"}|\text{"}live\text{" "}in\text{"}) = \frac{c(\text{"}live\ in\ New\text{"})}{c(\text{"}live\ in\text{"})} = \frac{1}{2} = 0.5,$$

$$\mathbb{P}(\text{"}York\text{"}|\text{"}in\text{" "}New\text{"}) = \frac{c(\text{"}in\ New\ York\text{"})}{c(\text{"}in\ New\text{"})} = \frac{1}{1} = 1.$$

Replacing those values in the equation, the result is:

$$\mathbb{P}(\text{"}They\ live\ in\ New\ York.\text{"})) = 1 * 0.5 * 1 = 0.5.$$

The n-gram probabilities are modeled from a training corpus, which makes the corpus a very important subject of study. Using a corpus too specific for the calculations may lead to failing to generalize the n-grams for new sentences. On the other hand, if the corpus is too broad, the n-grams may not capture the specificity of the domain one might be interested in modeling [JM00].

Even though language modeling through n-grams is a somewhat simple process, there are details that need to be adjusted depending on the context. Punctuation is one such example. Whether punctuation should be represented in the n-grams, or simply ignored, is highly dependent on the context. Author-identification and spelling error detection are just some of the examples where punctuation is fundamental to the process. The decision of whether the capture of n-grams

[1] Some important details are ignored for the sake of simplification.

should be done in a case sensitive fashion, or not, is yet another important detail that needs to be accounted for. The representation of each word form of the same abstraction, separately or together, as for instance `cat` and `cats`, should also be considered given the resulting impact of such change [JM00].

2.2.2 Smoothing

The fact that the n-grams are calculated from a finite corpus results in many cases of n-grams which do not get represented, thus obtaining a probability of zero when they should not. This problem is called data sparsity.

Smoothing is a process used to assign a value to some of the n-grams with zero probability, re-evaluating and adjusting the probability of others with low and high incidence [JM00]. The simplest case of smoothing is the **Add-One smoothing**, which is the addition of one to the count of each n-gram in order to prevent n-grams from having zero probability. With these changes applied to Equation 2.4 the result is:

$$\mathbb{P}_{add}(w_i|w_{i-n+1}^{i-1}) = \frac{1 + c(w_{i-n+1}^{i})}{|V| + \sum_{w_i} c(w_{i-n+1}^{i})}, \quad (2.5)$$

where w_{i-n+1}^{i} are all the words from the n-gram. Adding one to the count of each n-gram would result in an increase of the total probability of n-grams, which is balanced by adding $|V|$ in the denominator. V denotes the vocabulary containing all the words that should be considered, and $|V|$ is the number of words in the vocabulary. Once again, considering that Example 1 is the corpus, using the ML to estimate the probabilities of the 2-grams `in live` and `in middle` would result in 0, while using Equation 2.5:

$$\mathbb{P}_{add}(\text{"}live\text{"}|\text{"}in\text{"}) = \frac{1 + c(\text{"}in\ live\text{"})}{|V| + c(\text{"}in\text{"})} = \frac{1+0}{19+3} = \frac{1}{23} = 1/22 \text{ and}$$

$$\mathbb{P}_{add}(\text{"}middle\text{"}|\text{"}in\text{"}) = \frac{1 + c(\text{"}in\ middle\text{"})}{|V| + c(\text{"}in\text{"})} = \frac{1+0}{19+3} = \frac{1}{22} = 1/23.$$

This particular case of smoothing is actually worse than not using smoothing at all [JM00], mainly due to the fact that adding one to each n-gram represents a significant change in the mass, removing weight from the most important cases.

When the existing information on some n-gram is insufficient, resorting to lower order models might provide useful information on the higher order models [CG96]. Following upon this idea, some smoothing techniques use, for instance (n-1)-grams to help estimate the probability of some n-grams. This can be applied recursively, using the probability of (n-2)-grams to help estimate the probability of (n-1)-grams, and so on. There are two ways of resorting to this n-gram hierarchy [JM00], interpolation or backof. In the case of interpolation, the information

from the hierarchy is always used to estimate the probability of an n-gram as follows:

$$\mathbb{P}_{interp}(w_i|w_{i-n+1}^{i-1}) = \lambda\mathbb{P}(w_i|w_{i-n+1}^{i-1}) + (1-\lambda)\mathbb{P}_{interp}(w_i|w_{i-n+2}^{i-1}). \quad \textbf{(2.6)}$$

Interpolation continues until it reaches the unigram case, which is directly calculated from its probability. The value of $\lambda \leq 1$ is obtained from the training on held out data. Instead of using a single value for all n-grams, a specific value of λ for each (n-1)-gram ($\lambda_{w_{i-n+1}^{i-1}}$) could provide a more refined calculation of the probability. However, training each individual value would require a large amount of data and, as such, in most cases, *buckets* are created, attributing the same values of λ to groups of n-grams.

As for backof, it only resorts to the hierarchy when the count for the n-gram is zero, which results in:

$$\mathbb{P}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \mathbb{P}(w_i|w_{i-n+1}^{i-1}), & if\ \mathbb{P}(w_i|w_{i-n+1}^{i-1}) > 0 \\ \alpha_1\mathbb{P}(w_i|w_{i-n+2}^{i-1}), & if\ \mathbb{P}(w_i|w_{i-n+1}^{i-1}) = 0 \\ & \quad and\ \mathbb{P}(w_i|w_{i-n+2}^{i-1}) > 0 \\ & \vdots \\ \alpha_{n-1}\mathbb{P}(w_i), & otherwise, \end{cases} \quad \textbf{(2.7)}$$

where the α values ensure that the probability distribution does not sum to more than one.

Jelinek-Mercer smoothing is the simple application of interpolation to the ML. Considering once again that Example 1 is the corpus, one can estimate the probabilities of the same 2-grams `in live` and `in middle`, using Equation 2.6 with, for example $\lambda = 0.5$:

$\mathbb{P}_{JM}(\text{"}live\text{"}|\text{"}in\text{"}) = \lambda\mathbb{P}_{ML}(\text{"}live\text{"}|\text{"}in\text{"}) + (1-\lambda)\mathbb{P}_{JM}(\text{"}live\text{"}) = 0.5*0 + (1-0.5)*\frac{3}{26} = 3/52$ and

$\mathbb{P}_{JM}(\text{"}middle\text{"}|\text{"}in\text{"}) = \lambda\mathbb{P}_{ML}(\text{"}middle\text{"}|\text{"}in\text{"}) + (1-\lambda)\mathbb{P}_{JM}(\text{"}middle\text{"}) = 0.5*0 + (1-0.5)*\frac{1}{26} = 1/52,$

unlike the results using Add-one, this time `in live` receives a higher probability than `in middle` even without any of the sequences ever appearing on the corpus. This is a reflection of the higher number of occurrences of `live` when compared to `middle`.

Absolute discounting also uses interpolation, but instead of multiplying the probability of the n-gram by λ (or $\lambda_{w_{i-n+1}^{i-1}}$) it subtracts a fixed discount $D \leq 1$ from each non zero count, resulting in:

$$\mathbb{P}_{abs}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^{i}) - D, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + (1 - \lambda_{w_{i-n+1}^{i-1}})\mathbb{P}_{abs}(w_i|w_{i-n+2}^{i-1}), \quad \textbf{(2.8)}$$

in which $1 - \lambda_{w_{i-n+1}^{i-1}}$ is calculated from:

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\bullet), \quad (2.9)$$

to ensure that the sum of the distribution is one. $N_{1+}(w_{i-n+1}^{i-1}\bullet)$ is the number of unique words that follow the left context (or history (w_{i-n+1}^{i-1})) of the n-gram, formally defined as:

$$N_{1+}(w_{i-n+1}^{i-1}\bullet) = |\{w_i : c(w_{i-n+1}^{i-1} w_i) > 0\}|, \quad (2.10)$$

where the N_{1+} denotes the number of words that have one or more counts, and the $\bullet$ a variable that is summed over.

The value of D in Equation 2.8 is calculated from:

$$D = \frac{n_1}{n_1 + 2n_2}, \quad (2.11)$$

where n_1 and n_2 denote the total number of n-grams with the count of one and two, respectively, in the training data.

Kneser-Ney smoothing is an extension to the absolute discounting, based on the idea that the probability of an n-gram should not be proportional to the number of occurrences, but proportional to the number of words it follows. Kneser-Ney smoothing formula uses backof and it can be calculated by:

$$\mathbb{P}_{KN}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{\max\{c(w_{i-n+1}^{i}) - D, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})}, & if\ c(w_{i-n+1}^{i}) > 0 \\ \gamma(w_{i-n+1}^{i-1}) \mathbb{P}_{KN}(w_i|w_{i-n+2}^{i-1}), & if\ c(w_{i-n+1}^{i}) = 0 \end{cases}, \quad (2.12)$$

where $\gamma(w_{i-n+1}^{i-1})$ is chosen to make the distribution sum to one, using the right hand side from Equation 2.9. For the case of the unigram:

$$\mathbb{P}_{KN}(w_i) = \frac{N_{1+}(\bullet w_i)}{N_{1+}(\bullet\bullet)}, \quad (2.13)$$

where $N_{1+}(\bullet w_i)$ is the number of words that appear in the corpus before the word w_i, and $N_{1+}(\bullet\bullet)$ being the number of bigrams in the corpus. If Example 1 is the corpus and one wants to estimate the probability of the 2-grams `live San` and `live Francisco` using Equation 2.13, and given that neither of them appears the corpus ($c(w_{i-n+1}^{i}) = 0$):

$$\mathbb{P}_{KN}(\text{"}San\text{"}|\text{"}live\text{"}) = \gamma(\text{"}live\text{"})\mathbb{P}_{KN}(\text{"}San\text{"}),$$

where:

$$\gamma(\text{"}live\text{"}) = \frac{D}{c(\text{"}live\text{"})} N_{1+}(\text{"}live\text{"}\bullet),$$

where:

$$D = \frac{n_1}{n_1 + 2n_2} = \frac{19}{19 + 2 * 2} = \frac{19}{23} = 19/23,$$

with n_1 and n_2 being the number of bigrams with count one and two, respectively. For the calculation concerning the unigram, one should use:

$$\mathbb{P}_{KN}(\text{"}San\text{"}) = \frac{N_{1+}(\bullet\text{"}San\text{"})}{N_{1+}(\bullet\bullet)} = \frac{2}{21} = 2/21.$$

Wrapping up everything:

$$\gamma(\text{"}live\text{"}) = \frac{19/23}{3} * 2 = \frac{19 * 2}{23 * 3} = 38/69,$$

and finally:

$$\mathbb{P}_{KN}(\text{"}San\text{"}|\text{"}live\text{"}) = \gamma(\text{"}live\text{"})\mathbb{P}_{KN}(\text{"}San\text{"}) = (38/69)(2/21) \approx 0.052.$$

For the `live Francisco` 2-gram the calculation is similar:

$$\mathbb{P}_{KN}(\text{"}Francisco\text{"}|\text{"}live\text{"}) = \gamma(\text{"}live\text{"})\mathbb{P}_{KN}(\text{"}Francisco\text{"}),$$

where:

$$\mathbb{P}_{KN}(\text{"}Francisco\text{"}) = \frac{N_{1+}(\bullet\text{"}Francisco\text{"})}{N_{1+}(\bullet\bullet)} = \frac{1}{21} = 1/21,$$

resulting in:

$$\mathbb{P}_{KN}(\text{"}Francisco\text{"}|\text{"}live\text{"}) = \gamma(\text{"}live\text{"})\mathbb{P}_{KN}(\text{"}Francisco\text{"}) = (38/69)(1/21) \approx 0.026.$$

The only difference between the two n-grams is the second word and both words appear the same number of times in the corpus. Nonetheless, `San` is the successor of two different words while `Francisco` only follows one (`San`), justifying the obtained values.

Chen and Goodman [CG96] introduced a **modified version of Kneser-Ney smoothing** with three major differences:

- interpolation is used instead of backof;
- three discounts are used, one for counts of one, another one for counts of two and the third one for every other counts;
- discounts are estimated on held out data, instead of using a formula based on the training data (as in Equation 2.11).

Not all methods of smoothing were presented in this subsection. Instead, it focused on those

which allow for a better understanding of the modified version Kneser-Ney smoothing provided that it is the one with the best performance [CG96]. Nonetheless, there is another version of the Kneser-Ney modified, in which the discounts are once more calculated from the formula used in the original algorithm on the training data, avoiding the optimization of these parameters with only a slight drop in performance [CG96].

There are other techniques with the purpose of solving the problem with sparse data besides smoothing. However, techniques such as word classing and decision-tree models assume the use of language models different from n-grams [CG96]. Given that the most common language model and the one used in this work are n-grams, these methods will not be further described or explained.

2.2.3 Performance Evaluation

Evaluating the performance of a language model means measuring how well the computed model represents the data under analysis. The most common metrics used for evaluating the performance of a language model are the probability, cross-entropy and perplexity. They are usually calculated on a set of held out test data [CG96]. The probability of a set of data is simply the product of the probability of all sentences in the set.

The cross entropy can be measured using:

$$H_p(T) = -\frac{1}{W_T} \log_2 p(T), \quad (2.14)$$

where W_T is the number of words of a text T and the result can be interpreted as the average number of bits need to encode each word from the test data.

As for the perplexity ($PP_p(T)$), it can be calculated using Equation 2.15:

$$PP_p(T) = 2^{H_p(T)}. \quad (2.15)$$

Models with lower cross entropies and perplexities are better. Depending on the type of text, cross entropies can be between 6 and 10 bits/word for English texts, corresponding to values of perplexity between 50 and 1000. [CG96].

2.3 Topic Modeling

Topic modeling algorithms are statistical methods that are used with the objective of finding the subjects (or topics) presented in a collection of documents [Ble12]. These algorithms resort to the words of each document, and the topics at which they are most commonly associated

with. For this reason, the analysis usually ignores the words belonging to a stop-words list [2]. With the analysis complete, the result is a distribution over the topics for each document in the collection. Even though the topics associated with each document are the same, the differences between the probabilities in each distribution allow for a characterization of each document by its most probable topics.

The resulting distribution over the topics is a representation of the document in what is known as the *latent semantic space*. This representation presents a dimensionality reduction, when compared to the term *frequencies vector*, which is capable of more easily capturing the differences and similarities between documents in a collection [Hof99b]. The idea "is that documents which share more frequently co-occurring terms will have a similar representation" [Hof99b] (distribution), even if they have no terms in common.

Topic modeling is used in this work with the objective of finding similar documents to the one being written, and then using their content to calculate suggestions of improvements.

2.3.1 Plate Notation and Terminology

Topic modeling formally defines: a *word* as the basic unit of data, an item from a vocabulary; a *document* as a sequence of N words; and a *corpus* as a collection of D documents [BNJ03].

Plate notation is a graphical model for simplifying the process of representing variables that repeat themselves and their interdependencies (example in Figure 2.1). Each rectangle (or plate) groups a set of variables (circles) that are repeated, in the same context, a predefined number of times (in the case of both $Z_{d,n}$ and $W_{d,n}$, at Figure 2.1, they are repeated N times). The color of the circle represents the visibility of the variable: white circles are hidden variables, while gray circles are observed variables. Each link that connects two variables represents a dependency. For instance, in Figure 2.1 $W_{d,n}$ depends on $Z_{d,n}$ and $Z_{d,n}$ depends on I_d. When a link crosses the border of a plate it means that the variable on the outside is connected to each of the instances of the variable on the inside (I_d connects to each $Z_{d,i}, i \in [1, N]$).

2.3.2 Generative Probabilistic Topic Modeling

Generative probabilistic topic modeling is a group of algorithms that find topics by considering that each document in the collection is created by a process called the generative process. This process considers the existence of a latent structure, also known as hidden. The objective of this set of algorithms is then to reconstruct the structure, resorting to the observed variables which are, in most cases, the words of each document in the collection. As for the hidden structure, it is composed of (latent) variables that vary from model to model but, that generally include a probability distribution over topics, when the model considers each document a mixture of topics, representing the possibility of each document depicting more than one topic. This pro-

[2] Stop-words are words which contain very little topical information (*e.g.* `and`, `or` and `for`).

cess usually assumes that the topics exist prior to their execution, including their probability distribution over the words.

The generative model associated with **Probabilistic Latent Semantic Indexing (PLSI)** is called the *aspect model* [Hof99a, Hof99b] and is represented in Figure 2.1. The generation of each word in a document, according to the asymmetric formulation of this model, starts with the selection of a document d, with index I_d in the collection, with a predefined probability inside the collection. Then, for each word token [3], a latent class $Z_{d,n}$ is chosen from the probability distribution of the document over the latent classes and, according to the probability distribution over the words in that class, a word $W_{d,n}$ is finally selected. This process is repeated for each of the documents in the collection.

As represented in Figure 2.1, the fact that I_d is an observed variable means that the model considers only documents in the analyzed collection, and thus is unable to determine the topics for a new document afterwards.

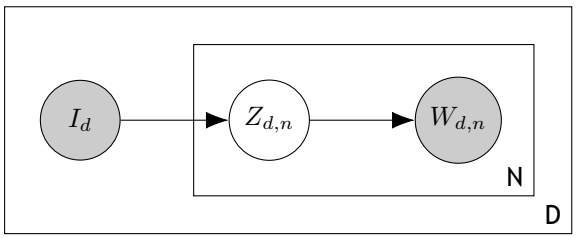

Figure 2.1: Graphical model representation of PLSI.

This process is based on three main assumptions: each word is considered independent from the others when conditioned on the topic assignment, describing each document as a *bag-of-words*, in which the order of the words is ignored; considers that words conditioned on the topic assignment are independent of the document in which they insert themselves; the number of existing topics is considered known and fixed [BNJ03]. All of these assumptions simplify the process of recovering the latent structure, which would be infeasible otherwise.

The **Latent Dirichlet Allocation (LDA)** (Figure 2.2) considers a generative process with complementary parameters to those presented in the PLSI process with the addition of two new parameters: α and β. α serves as a configuration parameter for the Dirichlet distribution, determining the distribution of topics for each document. On one hand, a small value of α is responsible for promoting distributions that have few topics with high probability. On the other hand, a high value of α promotes a high number of topics with identical probabilities. Similarly, β is a configuration parameter for the Dirichlet distribution over words. A small value of β means that each topic will describe few words with high probability, while a high value would describe a big number of words with comparable probabilities [GS04]. Usually, the objective is

[3] A representation of the position of a word in the document.

to have both a reduced number of topics as well as a reduced number of words in each topic.

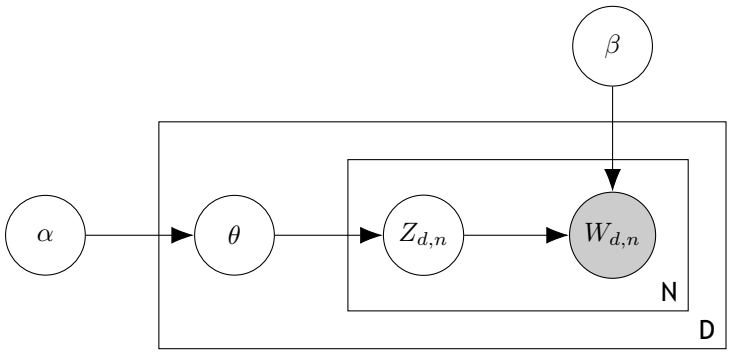

Figure 2.2: Graphical model representation of LDA.

Another noticeable difference is the exchange of the observed I_d for an unobserved θ_d. This change shows that, unlike PLSI, the LDA is capable of calculating a distribution over topics for new documents (outside of the collection), since it does not depend on a given index. θ_d is the topics distribution for a given document d.

Hierarchical Dirichlet Processes (HDP) is a nonparametric Bayesian model for clustering problems, where nonparametric colloquially represents that the number of clusters (topics, in this case,) is open ended [TJBB05]. This means that, unlike LDA and PLSI that consider the number of topics to be fixed and defined by the user, HDP is capable of updating the number of topics according to the data. The number of topics can even increase with new documents, which can be helpful when analyzing a growing and changing collection [BCD10].

The generative process (represented in Figure 2.3) begins with the selection of the base distribution G_0 from a Dirichlet Process (DP) (with H and γ). Each document is generated by first selecting the topics distribution G_d from a DP (with G_0 and α) and then the topic $\beta_{d,n}$ for each word token. Finally, the word $W_{d,n}$ from that topic is selected [BCD10].

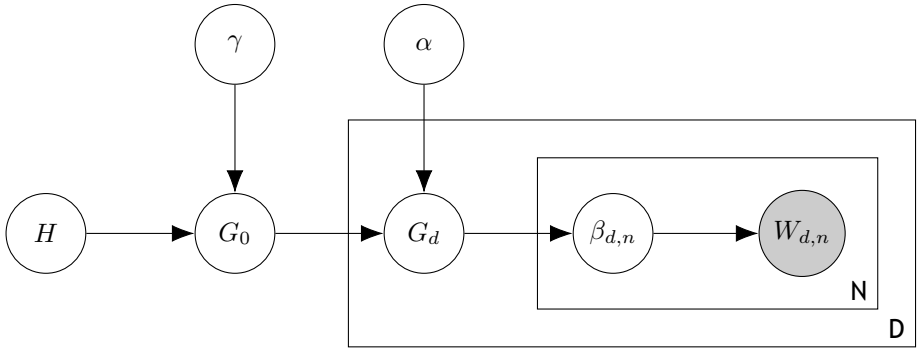

Figure 2.3: Graphical model representation of HDP.

According to its authors [TJBB05], HDP showed similar results to the ones presented by LDA when running with the best possible number of topics. This presents a clear advantage towards

HDP given that the user no longer needs external methods for discovering the number of topics while continuing to achieve the best possible results.

Topical N-Gram (TNG) is a model that combines topics with n-grams, allowing for the context to influence the results, rather than considering each word token an independent variable, as in the *bag-of-words* assumption taken by other models. Through the analysis on the context, each word token is either considered an unigram, which means it is independent from the context, or a bigram, uniting it to another word, in order to fully capture their topic. `white house` is a good example of this [WMW07]. Depending on the context, it can either be related to politics and have a special combined meaning, or simply be a description of a house in a real estate article, that can be separated without the loss of meaning. N-grams with order higher than two (bigrams) are possible by concatenating consecutive bigrams. The authors consider that the results show that their model is easier to interpret than the LDA due to the combination of word tokens.

The models described up to this point include the ones more commonly used for the topic modeling of documents and those closely related to this work. The following descriptions are introductions to other proposals of models.

The *author-topic* model proposed by Michal Rosen-Zvi *et. al.,* in [RZGSS04] is an extension to the LDA that introduces authors. The objective of this model is to simultaneously capture the topics of a document as well as the interests of an author. The *composite model* was proposed by Griffiths *et. al.,* in [GSBT05] and its main characteristics include the distinction between content and function words and the enforcement of syntactic structure for the generation of the document. The model proposed by Cohn and Hofmann [CH01] is an extension to the PLSI model where the topics assigned to a document are also dependent on the hyperlinks and citations existing in the document.

2.3.3 Estimation methods

The objective of generative probabilistic topic modeling is to obtain the topics that define a document, by calculating the latent structure defined in the generative model. Since the number of variables is usually big, the posterior distribution of the latent variables becomes intractable for exact inference, and thus, one needs to resort to other methods for estimation and inference. Some methods for estimation are introduced in this subsection.

According to the literature, namely Hofmann in [Hof99a, Hof99b], the standard procedure at that time (1999), for the estimation of the maximum likelihood in latent variable models was the Expectation Maximization (EM) algorithm. This algorithm is constituted by two steps, a step (E), where the posterior probabilities are calculated, which represent the probabilities of each topic being assigned to a given word, based on current parameters; and a maximization

step (M) where these parameters are updated. However, this estimation could not be directly interpolated for documents outside of the collection. As a solution for this problem and as a method of preventing overfitting towards the documents in the collection, those same papers proposed Tempered Expectation Maximization (TEM). The results reported that the usage of this algorithm led to increased precision and recall in the test data.

Griffiths and Steyvers showed that topic modeling with *Gibbs Sampling* was possible and a viable alternative to the previously proposed methods, in terms of both speed and memory usage [GS04]. The Gibbs Sampling algorithm utilizes a Monte Carlo Markov Chain (MCMC) for estimating the topic assignments and then infer the distribution over words in a topic and the probability distribution over topics for each document. The MCMC updates the values for the latent variables at every interaction by conditioning on the previously calculated values for the very same variables, until it starts converging to the target distribution.

2.3.4 Number of Topics and Evaluation

The number of topics is considered, by many algorithms, to be a fixed number. As such, external methods were required for helping in the choice of such number in the scope of this work.

The classic approach is to choose the number of topics that yields the highest probability for a test set of held out data. The user starts by dividing the documents into two sets: a train set where the topic modeling is performed; and a test set for the calculation of probability of the resulting topic distributions. Then, by experimenting on a range for the number of topics and then performing tests on the test set, the user is capable of discovering the number of topics resulting in the highest probability.

There is a large set of methods for performing the calculation of the probability on the held out data but, according to the study presented in [WMSM09], some provide more accurate results than others. The study analyses some importance sampling methods, namely the harmonic mean method, the annealed importance sampling and two new methods proposed by the author. The author concludes that both Chib-style method and left-to-right method produce more accurate results and should be the ones used in most cases. As an alternative, Griffiths and Steyvers proposed a method for the calculation of the number of topics in an automatic fashion [GS04]. As described in Subsection 2.3.2, HDP (and other nonparametric models) solve this problem by updating the number of topics according to the train data.

2.3.5 Document Comparison

The similarity of documents can be measured by comparing their topic distribution. A standard function to measure the difference between two distributions (p and q) is the Kullback Leibler

(KL) divergence [SG], given by:

$$D(p,q) = \sum_{j=1}^{T} p_j \log_2 \frac{p_j}{q_j}. \quad (2.16)$$

In many applications it may be convenient to apply a symmetric measure based on the KL divergence [SG]. In these cases, one can use:

$$KL(p,q) = \frac{1}{2}[D(p,q) + D(q,p)]. \quad (2.17)$$

Other alternatives include Jensen-Shannon divergence and, considering that these topic distributions are vectors, Euclidean distance, dot product and cosine.

2.4 Scientific Text Standards

Since the main objective of this master's project is the development of a set of algorithms capable of improving the quality of scientific documents, this section presents some of the studies performed on this subject. The studies focus on the structure of scientific articles and purpose of each section, on the analysis of the quality of a document and on tools developed with the objective of assisting in the writing of this type of documents.

2.4.1 Structure

IMRAD is the structure used in many research articles and, as suggested by the name, it refers to scientific documents which follow the structure: Introduction, Methods, Results and Discussion. An important concept in this subject is the concept of `move`, which refers to a "text segment with a specific rhetorical value" [Rei06].

In the context of IMRAD structure, Swales pioneered the analysis on the moves that constitute the `Introduction`, presenting the Create A Research Space (CARS) model. According to the moves determined by Swales (and explained in [Sut00]), the `Introduction` should include: (i) "the significance and centrality of the research area" [Sut00]; (ii) a discussion on some examples of previous studies and related work; (iii) the specification of a gap in the related work, justifying the proposed study; and (iv) a brief description of the study (which might be left implicit), including how it solves the aforementioned problem. Example 2 was extracted from [Sut00] and it helps in providing a better intuition on each move. The first sentence describes the centrality of the research area (move one). The next three sentences describe previous studies (second move) and the fifth sentence specifies a problem that was missed by the related work (move three).

Example 2. Many writers have noted the conflict between idealism and reality in Athol Fugard's "Master Harold" …and the Boys. Dennis Walder, for example, describes a "gap between the […] harsh, even violent reality" the play's characters endure and the "ideal world imagined by Sam" with his "idea of dancing as a paradigm of universal harmony" (122). Others have noted a second, closely related conflict, that between self-esteem and self-loathing. Frank Rich observes, "Fugard's point is simple enough: before we can practice compassion […] we must learn to respect ourselves" (C21). But no writer has pointed out that both conflicts are neatly summarized within the play by one more conflict: that between looking up and looking down. (120)

In 1997, Nwogu [Nwo97] proposed a model with modifications to Swales moves and with the addition of moves for the remaining sections of a document (following an IMRAD organization). This model can be summarized as follows, skipping the previously discussed `Introduction`: the `Methods` section generally includes a description on the data and method used for recovering it, on the experimentation performed and on the data analysis and classification; the `Results` section identifies the observations, distinguishing between successful and unsuccessful observations; finally, the `Discussion` section highlights the outcome of the research, explains the results obtained and presents the conclusions [Nwo97].

Swales, and then Ping *et. al.,* [HZW10], point out that the importance of each move is highly dependent on the discipline. Some moves might not be used in certain disciplines, while others are used more than once, as is the case of the second move, which can sometimes be used in a cyclic fashion, for instance in *Computer and Security* and *Computer Languages* [HZW10].

Louvigné *et. al.,* [LS15] focused on the `Results` section, concluding that it is closely related to the `Methods` section, resulting in the use of related moves. According to the authors, this happens mostly to improve the connection between the methods proposed and their findings, better justifying the results achieved [LS15].

Figure 2.4 presents a representation of this structure [GD10], considering the *moves* of each section of the IMRAD organization. The representation is vertically symmetrical because the `Discussion` will serve some of the same purposes of the `Introduction`, in reverse order. [GD10] justifies this fact with two examples: the first is the need for a way to start and a way to end the document; and the second one is the need for the creation of an interface with the central section (`Methods` and `Results`). Another important detail presented in Figure 2.4 is that the central report section is narrower, getting widener as it gets further away from this section. This concerns the general tone used in the beginning of the `Introduction`, which gradually focuses on the specific subject detailed in the document, and the opposite is true for the `Conclusion` [Rei06].

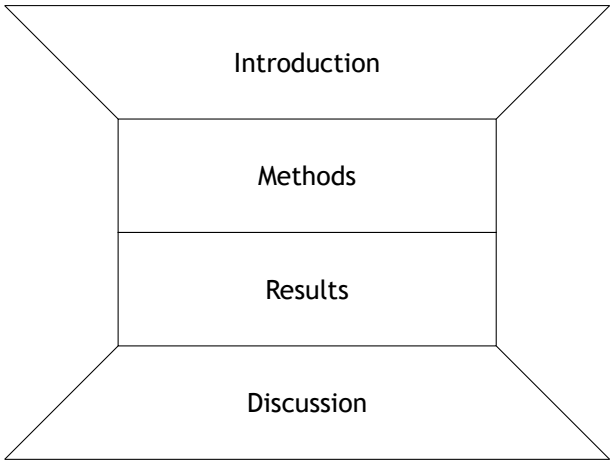

Figure 2.4: Representation of the IMRAD organization.

2.4.2 Quality

Well written texts are described by characteristics that go beyond the correct use of grammar, correct spelling and organization of the document [LN13]. A well written story is characterized by different factors than the ones taken into account when studying a good scientific publication.

Specificity of the text and communicative goals are the two characteristics studied in [LN13], in the context of science journalism with a parallel to scientific documents. The specificity of text refers to the hypothesis that a correct balance between providing general high level information and very specific details can contribute to the text quality. The communicative goals refer to the intentions of each segment in the text, and to the order in which they appear. These communicative goals are similar to the definition of move described in Subsection 2.4.1.

The writing of articles containing a mixture of general and specific statements is a good formula to retain the attention from the reader, delivering the message with increased clarity [LN13]. As for scientific publications, documents usually follow a very thorough and well defined structure in terms of specificity of statements, presenting general content at the beginning and end, and detailed content in between. Example 3, extracted from [LN13], shows an example of a general sentence (first sentence) and a specific sentence (second sentence).

> Example 3. Dr. Berner recently refined his model to repair an old inconsistency. [general] The revision, described in the May issue of The American Journal of Science, brings the model into closer agreement with the fact of wide glaciation 440 million years ago, yielding what he sees as stronger evidence of the dominant role of carbon dioxide then. [specific]

Given that the organization of intentions at low levels contribute to the coherence of the overall conveying of purpose of a document, analyzing them constitutes a method for determining the quality of a text. In scientific documents the communicative goals include aim, background, results, among others [LN13].

Automatic Essay Scoring (AES) is the use of computer technology for the automatic evaluation and scoring of written prose [Dik06], which provides a good source for analyzing the quality of a document. Project Essay Grade (PEG) is one example of an AES system and the idea behind its scoring was to use indirect measures (referred to as proxy measures [COJ97]) to estimate a specific quality in the writing of the essay. The example given in [COJ97] is that of diction (which is the appropriate words choice) is measured via estimation of the proportion of uncommon words in an essay [COJ97]. The PEG is first trained with human rated essays, adjusting its weights according to the measures recovered from the essays and the grades provided by the human raters.

2.4.3 Tools

This subsection presents three subsets of tools related to the study of scientific documents. The first subset is constituted of tools that provide general information helpful in the writing or analysis of this sort of documents. The two tools that follow are included in this category of tools:

- Thesis Writer [RKEO15] is a support system developed by Christian Rapp *et. al.,* with the goal of aiding in the writing of a thesis. It provides support tools and tutorials for every phase of the process, from the conception to the completion of the thesis. It was designed with collaboration in mind, allowing tutors, instructors and other students to provide feedback directly in the online text editor;

- Duygu Simsek *et. al.,* [SSDL+14] describe a new tool called Xerox Incremental Parser (XIP), which is capable of automatically highlighting metadiscourse markers. These markers are present in every document in research articles, and point to the introduction of key arguments. The tool labels certain sentences inside a document, according to the type of function it provides: summary, novelty, contrasting ideas, and others. By analyzing the labels, users can more easily understand and study research articles.

The second subset of tools provides information specific to the document being written in but without proposing automatic solutions, and includes:

- O'Rourke *et. al.,* [OC09] proposed a method for visually representing the flow of paragraphs in a document. The process starts with the modeling of the topic mixture for each paragraph in the document. This mixture is then reduced to a two dimensions space, allowing for the visual representation of each paragraph in the document. The authors defend that, by visualizing the positioning of each paragraph in space, and the distance between them, the user should be capable of more easily uncovering problems in the flow of ideas;

- OpenEssayist [WTR+15] is an analytics tool that provides automatic feedback on an essay. Users can access its web application, submit their essays and get feedback on summaries, keywords, words distribution and a collection of other statistics. According to the reported analysis, students who used the system more frequently had a tendency to obtain better results;
- Masaki Uto *et. al.,* [UU15] propose a system of support in the writing of arguments for academic documents. It consider the difficult inherently associated with the writing of arguments in an academic context, namely in long arguments, as the motivation for the developed system. The system uses a Bayesian Network representation of the Toulmin model, which is the standard model for evaluating an argument. They concluded, through subjective testing, that the system is indeed capable of supporting the elaboration of arguments.

The last set contains tools with a similar purpose to the one proposed in this dissertation, providing automatic suggestions for improvements or replacements. The tools include:

- Jian-Cheng Wu *et. al.,* [WCMC10] developed a system capable of providing suggestions for English academic texts. A classifier was trained with the verb-noun collocations, and their context information was extracted from a corpus of abstracts of published articles using machine learning. Every time a new sentence is checked, the system parses the sentence and extracts the verb-noun collocation. The collocation is then analyzed and the system selects the most likely collocates as suggestions for that verb. The main limitation of this solution is the fact that it can only provide suggestions for verbs;
- In [NSH12], a study with the purpose of choosing the best approach in the suggestion of verbs, given the left context of the verb, was presented. An academic writing model was used to create a corpus of abstracts and a verb ranking. Resorting to this model and ranking, queries are performed and the results considered a success when the proposed verb is the original one. The various approaches include searching for the exact left context, expanded versions of the left context with wildcards and pronoun and noun tags, as well as their potential combinations. The results show that the best approach is to utilize the expanded versions that include the wildcards and pronoun and noun tags when the exact match is not found. Once again, the limitations of this solution include the fact that suggestions are restricted to verbs.

2.5 Conclusion

This chapter discussed concepts on the most important subjects related to the work under development. General language modeling concepts were introduced, covering the most widely-used language models, which are n-grams [CG96]. These allow for the capture of important

features in documents but need to account for the data sparsity problem. This problem in n-grams is mainly solved by using smoothing techniques of which the best is the modified version of Kneser-Ney [CG96]. The state of the art then shifted to the field of topic modeling, introducing different methods for the extraction of subjects in a collection of documents and posterior comparison with documents outside of this collection. Of these arguable the most studied and used technique is the LDA [Ree12], mainly due to its simplicity [Ble12] but also for its overall good results. At last, given the main focus on assisting in the writing of scientific documents specifically, several characteristics of this type of document were studied and tools with similar purposes were presented. The study on this specific type of writing seems to show that even though there are some characteristics related to the evaluation of the quality of a document, there are no standards on the subject which makes it a hard task to perform.

This collection of knowledge should, hopefully, provide enough background for the complete understanding of the algorithm and its implementation presented in Chapter 3. Some of the aforementioned details were the basis for the selection of the methods implemented in the algorithm.

Chapter 3

Design and Implementation

3.1 Introduction

A study on the state of the art was presented in the previous chapter, introducing the main techniques related with the problem at hands. Considering the problem statement, the algorithm was designed with the objective of providing suggestions of replacements that are more commonly utilized in the area of knowledge of the documents under analysis. The algorithm can be understood as a combination of two sub-algorithms, as stated in the master's proposal: the first one is responsible for recovering the context of the document under analysis and finding documents under that context; and the second calculates suggestions for the document resorting to the documents selected by the first sub-algorithm.

The chapter begins with a high level description of the *Context and Similar Documents* sub-algorithm, which is based on the techniques presented in the previous chapter (Section 3.2). The chapter follows with a description of the corpus that was build in the context of the project and associated processes, in Section 3.3). It proceeds with the description on how the *Context Based Suggestions* sub-algorithm evolved, presenting the changes and reasons justifying those changes (Section 3.4). The subsequent section presents the tools utilized in the implementation of the algorithm (Section 3.5) and the chapter culminates with a complete walk-through over the algorithm, presenting examples for each of the phases (Section 3.6).

3.2 Context and Similar Documents

After the study performed on the state of the art, all the necessary techniques for the *Context and Similar Documents* sub-algorithm had been presented. This sub-algorithm (presented in Figure 3.1) is responsible for automatically extracting the context of a document in terms of its subjects (topics), and then selecting a set of similar documents for use in the calculation of suggestions. It begins with the processing of the corpus. Using a topic modeling algorithm, such as LDA, the hidden structure from the associated generative process is estimated (step 1), including a representation of the topic probability distribution for each document in the corpus. Using the same hidden structure, a topic probability distribution is estimated for the document under analysis (step 2). This distribution (representation) can then be compared with the distributions for each document in the corpus, using a method such as the KL so as to select its closest documents (step 3). Using the n-grams extracted from the corpus (step 4),

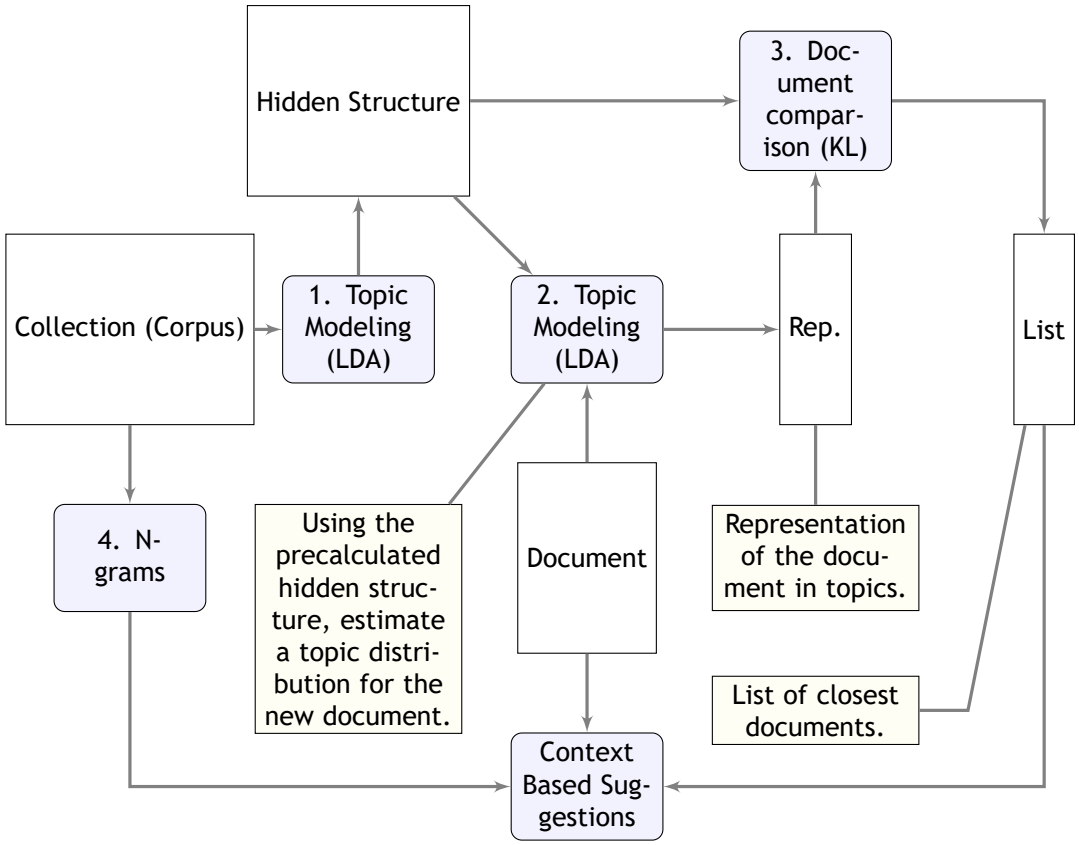

Figure 3.1: Context and Similar Documents sub-algorithm.

the document under analysis and the list of closest documents from the corpus, it should be possible to calculate suggestions according to the specific context of the document (*Context Based Suggestions* sub-algorithm). The second sub-algorithm is described further ahead in this section.

3.3 Corpus

The search for a corpus was one of the first steps of this work, preceding and accompanying the development of the algorithm. The corpus was necessary for feeding the topic model process, and as a baseline to be used later for the newly written documents.

The predefined requirements for the corpus resulted from the study of the state of the art. The main requirements identified for this corpus are the following:

- all documents must be scientific articles or papers published in journals or magazines. This requirement reflects the objective of providing assistance in the writing of scientific documents, thus the necessity for similar documents as a means of comparison and as a source of information. By resorting to previously published documents, a higher confidence that the documents to be used later as a reference have been thoroughly analyzed by scientific committees is achieved;

- the documents must be of at least two different areas of knowledge. Studying the specificity of the terminology used in different areas arose as an objective of the project. By performing tests using manuscripts from the area of the document under analysis, and then using manuscripts from other areas, and analyzing the results, one would expect that the existing differences between the areas would give rise to poorer results for the manuscripts from different areas. If similar results were achieved, then this could be interpreted as a sign that the identified areas have similar syntactic organization and terminology;

- the collection must include full documents and not only abstracts. The algorithm under development has to assist the author in the writing process of an article, and given the distinctive pattern [YXL10, YFL02] present in the abstracts, they are clearly insufficient to achieve that goal.

The search for a corpus with the desired characteristics proved to be more difficult than originally thought. Most of the corpus found were entirely formed by abstracts of scientific documents [KOTT03] or contained a single field of study [BDD+08]. There was one, however – SciTex [DOKLK+13] – which filled all the desired requirements. SciTex is constituted by two sub corpus, the SaSciTex, which includes scientific documents from the 1970s and early 1980s, and the DaSciTex, whose scientific documents are from the 2000s. These documents belong to categories including Computational Linguistics, Linguistics, Computer Science, Bioinformatics, Biology and Computer-aided design, among others. Nonetheless, SciTex was not used in the scope of this work because access is restrained to the direct interaction with Sketch Engine, which is a corpus manager. Even though corpus managers perform comprehensive analysis on the corpus, providing powerful tools and metrics for their study, the failure in the implementation of methods to call upon procedures from the Application Programming Interface (API) of this specific corpus manager, led to the pursuit of a new corpus.

After having no success in finding a suitable corpus, the alternative was to create a new one. The gathered corpus is composed of published scientific documents from an electronic archive, namely `arXiv`, whose main categories belong to: Physics, Mathematics, Computer Science, Quantitative Biology, Quantitative Finance and Statistics. Each category is subdivided into a varied number of subtopics. The documents were then chosen by following an approximation of an uniform distribution in respect to both the topics and each subtopic, even though limited to the number of existing documents. The download of the documents was performed through the execution of a Java script responsible for making a request for the information on a predefined number of documents from a discipline, parsing the response of `arXiv` and downloading each link obtained from the meta information. The script for downloading the files was entirely implemented during the course of this project.

The distribution of documents between the topics is shown in Table 3.1, in terms of number

Table 3.1: Document distribution for each area of knowledge in the corpus.

Area of Knowledge	Number of Documents
Physics	5051
Mathematics	5136
Computer Science	5321
Quantitative Biology	5000
Quantitative Finance	3977
Statistics	4163

of documents in each topic. The documents were then divided into training and testing sets: 80% of the documents from each subtopic were selected for training and the remaining for testing. Then, another division split 80% of the training set for training and the remaining 20% for validation. The validation set will be used to test different configurations, allowing for the fine tunning of the algorithm and consequent improvement of the results.

Given that the scientific papers from the corpus never change and the documents themselves are not needed, only their text, a set of preprocessing tasks were executed before using them. These tasks include the following:

1. Automatic parsing of each pdf document into a text document, resorting to the Apache PDFBox library [Fou10]. The resulting text documents contained parsing errors which, in some cases, resulted in splitting a word into multiple lines or adding spaces between characters. In order to attenuate this problem, all lines that had less than fifteen characters were removed from the parsed document. The number fifteen was chosen after an analysis on some of the resulting texts, by choosing the biggest number that could remove parsing errors, while retaining all the important sentences untouched;

2. Aggregation of all scientific papers from the training set into a single text document, where each line contains one of the original papers. This was the specific format required by the topic extraction tool;

3. Calculation of the n-grams for each document individually and their inclusion in the database. The first time the insertion of the n-grams in the database was tried, it took between two to ten minutes for each document, depending on the number of existing n-grams. With this latency, it would have taken more than a month just to insert the initial data into the database. It was found that the culprit was related with the fact that the table where the data was being inserted contained a `PRIMARY KEY` constraint, which involved the updating of an index each time a new row was inserted, slowing the entire process down. The solution was to remove the index and insert all the data, and only then recreate the index. With this procedure each document took less than a second to process and the insertion was complete after two hours. After the insertion was complete, the recreation of the primary key was tried, but failed with the report of duplicate entries. After checking some

of the cases that were flagged as duplicates, and noticing that even though with similar characters, the strings were not the same. This problem was then solved by changing the charset and collation for the fields in question, which determined the codification and rules used for storing and comparing data, respectively. The next step was then again the recreation of the primary key, which took another ten minutes.

The extracted n-grams resulted from the application of rules and restrictions that derived from the initial experiments on the data and from the study of the state of the art, specially from [JM00]. The applied rules are the conversion of each character to lower case, replacing each numeral [1] with a special tag and adding two special tags before the initial word and two special tags after the last word in each sentence (two special tags in order to form a complete trigram with the existing word). The restrictions applied to the data concern the exclusion of any character, besides punctuation, words and numbers, that breaks the sequence of n-grams. The whole process can be better understood by examining the n-grams in Table 3.2 obtained from Example 4. Notice that four extra n-grams were created due to the inclusion of the special tags signaling the beginning and end of the sentence. The restrictions also modify the results, removing all the n-grams with the symbol #.

Table 3.2: N-grams obtained from the Example 4.

N-gram	Count
[<string1>, <string2>, this]	1
[<string2>, this, is]	1
[this, is, a]	1
[is, a, simp]	1
[e, text, resulting]	1
[text, resulting, from]	1
[resulting, from, automatic]	1
[from, automatic, parsing]	1
[automatic, parsing, of]	1
[parsing, of, a]	1
[of, a, pdf]	1
[a, pdf, file]	1
[pdf, file, .]	1
[file, ., <stringn-1>]	1
[., <stringn-1>, <stringn>]	1

Example 4. This is a simp#e text resulting from automatic parsing of a pdf file.

The database was created using MySQL as Database Management System (DBMS) and the data organized into tables, whose name and purpose can be described as follows:

- 3gram, which contains a million of the most frequent n-grams from COCA. These n-grams represent generic English;

[1] A sequence of digits.

- `file_arxiv`, which stores the references to each document that constitutes the corpus;
- `cs`, containing the n-grams from Computer Science, with a pointer to the document of origin;
- `csTotal`, containing the n-grams from Computer Science, combining all equal n-grams, independently of the document of origin;
- `csCut`, which contains the n-grams from `csTotal` whose count is larger than one.

Additionally to the aforementioned tables, there are three other tables (similar to the ones from Computer Science) for each remaining topic.

3.4 Context Based Suggestions

This section presents the evolution of the method used for the calculation of the suggestions, alongside the reasons that led to the implementation of several changes.

3.4.1 Synonyms

The initial method for calculating suggestions resulted from a simple combination of tools, described in Section 3.5. The text was split into sentences using a `BreakIterator` and each sentence treated separately. Syntactic parsing was applied to each sentence using the Stanford Parser (Subsection 3.5.4) to tag the words from each sentence with their correspondent Part of Speech (POS) tags. The tags were then filtered as either adjectives, verbs, nouns or adverbs, and ignoring all other syntactic classes, since these provide no meaning on its own and, consequently, have no synonyms. N-grams were formed from every set of three consecutive words and synonyms of the last word of the n-gram were then obtained using Java Wordnet Interface (JWI) (Subsection 3.5.3). Each synonym was used to form a new n-gram, constituted by the two previous words and the synonym. The resulting suggestions came from the n-grams that had the highest chance of occurring, according to a database of n-grams. This can be though of as the base module (every node outside of other modules) in Figure 3.2 combined with module `B`.

The probability of the n-grams is calculated using the ML, with the backup probability of $1.0E{-}11$ being attributed to the cases when the probability was 0. The probability of the n-grams is dependent on the closest documents selected. This backup probability means that the probability distribution is incorrect (given that the remaining probabilities are not adjusted in accordance), but it attenuates for the non use of smoothing techniques.

The main problem associated with this approach was the small suggestion coverage. By restraining the words to a fixed set of syntactic classes (adjectives, verbs, nouns and adverbs), the suggestion coverage was vastly reduced, excluding possible replacements for other classes,

for instance between prepositions, as in `in` and `on` mistakes. In Example 5, the preposition `on` should be used instead of `in`, but with the aforementioned algorithm, only the words `did`, `example` and `Monday` were analyzed.

> Example 5. I did this example in a Monday.

3.4.2 N-grams

To fix the coverage problem affecting the initial method, a backup routine was implemented with the objective of obtaining a set of n-grams for the cases where the last word did not belong to the accepted syntactic classes. This function creates new n-grams for the ten most probable words that come after the two initial words of the n-gram and, once again, the most probable words are suggested as replacement. To improve the calculation of the most probable words, the words that occur after the word of interest are taken into account. The probability of each word is defined by the probability of the expression formed by the n-gram and the two next words (in case of trigrams), resulting in the combination of the probability of the three n-grams (in case of trigrams) in the expression. The words with the most likely expressions are selected as suggestions. This resulted in the addition of a simplified version of module `E`, in Figure 3.2, to the scheme described in Subsection 3.4.1.

3.4.3 Previous and Next Words

The version of the method described in Subsection 3.4.2 created a list of candidates for replacements, in the case that the word did not belong to the classes with synonyms on the Wordnet. Nonetheless, the choice of candidates did not include words which could occur with high probability before the next words and between the previous and next words. With this change performed, module `E`, in Figure 3.2, was complete. The remaining problem concerned the fact that the proposed suggestions for the accepted syntactic classes were in its base form, which is the only form of words existent in the Wordnet.

3.4.4 Morphological Realization

SimpleNLG was integrated to attenuate the base form problem. SimpleNLG is capable of calculating the inflected forms of a word given a set of defined features (morphological realization), including tense, number and person (in the case of verbs), among others. The integration of this tool allowed the algorithm to obtain the inflected form of some of the synonyms, resulting in the addition of module `C`, in Figure 3.2, to the scheme proposed in Subsection 3.4.3.

3.4.5 Prepositions

The last addition to the algorithm comprised the introduction of a particular treatment for the prepositions. Given that the most commonly used prepositions, in English, form a small list of words, whenever a preposition is analyzed, the prepositions from the list are all treated

as possible candidates for replacement. This change should improve upon the most frequent preposition mistakes, by providing a bigger set of alternative replacements. The addition of a special treatment for the prepositions is represented by module D, forming the complete scheme in Figure 3.2.

3.5 Used Tools

This section provides a succinct description for each of the tools that were integrated in the algorithm or used in the development of this work.

3.5.1 MALLET

The first component integrated into the algorithm was MAchine Learning for LanguagE Toolkit (MALLET), which is a "Java-based package for statistical natural language processing, document classification, clustering, topic modeling, information extraction, and other machine learning applications to text" [McC02]. MALLET is used in the algorithm for its topic modeling capabilities. A model of the corpus is initially created using LDA with the parameters $\alpha = 50/T$ and $\beta = 0.1$ (as suggested in [GS04]), where T is the number of topics. The number of topics was previously selected by calculating the probability of the model with $T \in [50, 1250]$, 50 by 50, in the validation set (as suggested in [GS04]). The results for Computer Science and Mathematics are represented in Figure B.1, highlighting the selected number of topics for each of the sets. With the model for the corpus calculated, MALLET is capable of calculating a topic probability distribution for the text being processed, using the same topics. Finally, through the comparison of the distribution of the document and the distributions from each document in the corpus, it is possible to identify the closest documents and use them as a source of information. Before extracting topics from a set of documents, all text is converted to lower case and stop-words are removed.

Table 3.3 shows five topics that were obtained from a total of 100 topics extracted using LDA, as an example of the topic modeling operation. Each topic represents a set of words that are usually associated with the same content. The example shows the probability distribution of each topic in the collection of physics documents from the corpus (Section 3.3) and the most common words from each topic. A quick analysis of the results suggests that the words from topic 0 belong to the *astrophysics* branch, while topic 2 presents generic words, not exclusively related to physics.

Table 3.3: Example of five topics extracted from the physics documents of the corpus.

Topic	Probability	Characteristic Words			
0	0,00626	mass	fomalhaut	disk	planet
1	0,03401	semileptonic	decays	phys	gev
2	0,02901	university	usa	united	kingdom
3	0,01339	structure	ring	atoms	surface
4	0,04978	phys	amplitudes	phase	decays

3.5.2 LanguageTool

LanguageTool represents a language-independent proofreading checker that allows a user to receive corrections to grammatical errors missed by most common spell checkers [Mił10, Nab03]. Errors are detected by checking existing rules and using dictionaries of the language of interest, creating a modular system that is constantly being expanded. The fact that it is open source allows for the analysis and verification of its functioning by the community. Using LanguageTool to analyze the sentence from Example 6 results in two suggestions: the first one concerns the fact that the first letter of the first word in the sentence is not a capital letter; and the second is related to the use of a plural word (`tests`) after an indefinite article (`a`).

Example 6. this is a tests.

3.5.3 Wordnet - JWI

The Massachusetts Institute of Technology (MIT) JWI [Fin14] constitutes an interface for the wordnet English electronic lexical database [Mil95]. This database contains a dictionary of words organized according to its semantic meaning. The database includes a definition of the associated concept, as well as relations between words such as synonymy, hyperonymy, hyponymy and antonymy, which represent associations between words referring to the same concepts, more generic concepts, more specific concepts and opposite concepts, respectively. All this information is made available by JWI, allowing for the retrieval of synonyms and creation of suggestions, based on the word and respective POS tag. As an example, using JWI to retrieve the synonyms for the word `skill` and the POS tag set to `NOUN`, results in the following list of words: `skill, accomplishment, acquirement, acquisition, attainment, science`. The main problem with the usage of this tool in the algorithm comes from the fact that all stored words are in its base form, thus becoming inadequate synonyms for most situations where the original word is in an inflected form.

3.5.4 Stanford Parser

The Stanford Parser groups a set of natural language parsers and models for some languages, including English, German, Chinese and Spanish. A natural language parser is a program responsible for splitting a text into sentences and finding the syntactic structure inherent to each one [KM03]. Example 7 shows the result of the application of Stanford Parser for POS tagging the sentence “Dr. Maboul is an experimental sentence.”. In the example,

Example 7. Dr./NNP, Maboul/NNP, is/VBZ, an/DT, experimental/JJ, sentence/NN, ./.

The utilized tags are defined in [San90] and include: Proper noun, singular (NNP); Verb, 3rd person singular present (VBZ); Determiner (DT); Adjective (JJ); and Noun, singular or mass (NN).

The developed algorithm uses this tool to obtain the syntactic function of each word in a sentence, represented by the POS tags. The POS tags are necessary for the processing performed by the JWI, with the objective of obtaining synonyms. They are also used to deduce extra information about some words, including number, person and tense, providing them to SimpleNLG.

3.5.5 Other Tools

The SimpleNLG is a realization engine for English capable of generating syntactic structures [GR09]. This engine is used to obtain the inflected form of a specific base form, in a given tense, person and number. By integrating this engine, the editor is capable of producing some suggestions in the appropriate form, instead of suggesting the base form.

The Apache PDFBox is an open source Java tool for working with pdf documents. It allows for the creation of new documents, and manipulation and extraction of content from the existing documents [Fou10]. This tool was used to extract text from scientific articles in the corpus, which were available as pdf documents.

3.6 Algorithm

Building up in the discussion included up to this point, the algorithm that resulted from the study performed is now presented. This section explains the entire process (represented in Figure 3.2), describing how each piece fits together to form the main algorithm, accompanied by examples with the outputs for each phase of processing. This section skips the treatment performed on the corpus before the execution of the algorithm since it was previously described in Section 3.2.

The algorithm was implemented in Java, with each of the examples deriving from the iterative application of the steps of the algorithm to the baseline in Example 8.

> Example 8. Generative probabilistic topic modeling is a group of algorithms that find topics by considering that each document in the collection is created by a process called the generative process.

3.6.1 Context and Similar Documents

Each analysis starts with the calculation of the topic probability distribution for the words of the entire document using LDA from MALLET (step 1). Given the probability distribution, the Kullback–Leibler distance is calculated for each document in the selected subset of the corpus. Those with a value inferior to a threshold (a parameter provided by the user) are selected and used in all queries for the computation of suggestions (step 2). These steps form the *Context and Similar Documents* sub-algorithm (module A in Figure 3.2), and they are ignored when the table utilized for the calculation of the suggestions contains the n-grams from the entire corpus

without a specification of the origin documents. The remaining steps are performed for each sentence instead, independently forming the *Context Based Suggestions* sub-algorithm.

3.6.2 Sentence Level

Using the Stanford Parser, the algorithm creates a tree structure representative of the syntactic structure of the sentence, attributing a POS tag to each word (step 3). Example 9 shows the output of this step of the algorithm for the sentence in Example 8. Even though a tree structure is created by Stanford Parser, the algorithm resorts only to the POS tags of each word.

> Example 9. Generative/JJ, probabilistic/JJ, topic/NN, modeling/NNS, is/VBZ, a/DT, group/NN, of/IN, algorithms/NNS, that/WDT, find/VBP, topics/NNS, by/IN, considering/VBG, that/IN, each/DT, document/NN, in/IN, the/DT, collection/NN, is/VBZ, created/VBN, by/IN, a/DT, process/NN, called/VBN, the/DT, generative/JJ, process/NN, ./.

Once again, the tags in the example are defined in [San90] and the new ones include: Noun, plural (NNS); Preposition or subordinating conjunction (IN); Wh-determiner (WDT); Verb, non-3rd person singular present (VBP); Verb, gerund or present participle (VBG); and Verb, past participle (VBN).

All the trigrams formed throughout the analysis of the sentence in Example 8 are shown in Table 3.4.

The remaining steps are performed for each word in the sentence under analysis. A word can be classified into two types of words: (i) content words, which are responsible for introducing the semantic content of the sentence and (ii), function words, which serve a syntactic function, providing no meaning by themselves (distinction defined in [GSBT05]). This classification is derived from the POS tag of the word.

3.6.3 Word Level

The first word in the sentence of Example 8 is `Generative` with `JJ` as the POS tag. This POS tag means the word is an adjective, which is one of the syntactic classes that represents a content word. Given that this is a content word, synonyms are gathered using the JWI (step 4). The list of synonyms provided for this word is presented in Example 10.

> Example 10. generative, productive, procreative, reproductive.

Since the tag JJ provides no information on the form of the adjective under analysis, `SimpleNLG` is unnecessary in this case.

Fast forwarding the analysis to the word `is`, it may be seen that it was tagged with `VBZ`, meaning

Table 3.4: N-grams obtained from the Example 8.

N-gram	Count
[<string1>, <string2>, generative]	1
[<string2>, generative, probabilistic]	1
[generative, probabilistic, topic]	1
[probabilistic, topic, modeling]	1
[topic, modeling, is]	1
[modeling, is, a]	1
[is, a, group]	1
[a, group, of]	1
[group, of, algorithms]	1
[of, algorithms, that]	1
[algorithms, that, find]	1
[that, find, topics]	1
[find, topics, by]	1
[topics, by, considering]	1
[by, considering, that]	1
[considering, that, each]	1
[that, each, document]	1
[each, document, in]	1
[document, in, the]	1
[in, the, collection]	1
[the, collection, is]	1
[collection, is, created]	1
[is, created, by]	1
[created, by, a]	1
[by, a, process]	1
[a, process, called]	1
[process, called, the]	1
[called, the, generative]	1
[the, generative, process]	1
[generative, process, .]	1
[process, ., <stringn-1>]	1
[., <stringn-1>, <stringn>]	1

that the verb form is the third singular person of the present. With this information, JWI recovers synonyms for `is` (presented in Example 11).

> Example 11. be, exist, equal, constitute, represent, make up, comprise, follow, embody, personify, live, cost.

The information provided by the `VBZ` tag is then utilized by `SimpleNLG` to perform the correct inflection of the synonyms (step 5). The resulting list is presented in Example 12 and it includes both the base and inflected forms of the synonyms.

> Example 12. is, be, exists, exist, equals, equal, constitutes, constitute, represents, represent, make up, make up, comprises, comprise, follows, follow, embodies, embody, personify, personify, lives, live, costs, cost.

The next word (`a`) is tagged as a `DT`, which means its a function word. Since it is not a preposition, no special treatment is performed on the word. Then comes the word `of` with the `IN` tag, meaning that it is a preposition. In this case, the entire list of prepositions is added to the list of candidate replacements (step 6).

In either of these cases, and similarly to the processing done for the word prediction, candidates for replacement are added based on the context of the word (step 7). The context n-grams utilized for the word `Generative` are [`<string1>`, `<string2>`, `*`], [`<string2>`, `*`, `probabilistic`] and [`*`, `probabilistic`, `topic`], where `Generative` has been replaced with a `*`. The `*` represents the position under analysis. The words that are most likely to appear at that specific location are added to the list of candidates. This step completes the collection of candidate replacements.

The final step is the selection of the most likely candidates (step 8). For `Generative`, for example, the same n-grams are used with the `*` being replaced by each of the candidates, and the combined probability of each n-gram with that candidate determines its chance of occurring. The user is then the entity responsible for either choosing to ignore or apply each suggestion.

3.7 Conclusion

The previous sections provided a description on the functioning of the algorithm. It started with a description of the sub-algorithm responsible for the extraction of the context and similar documents, followed by the discussion on how the corpus was build along with its main characteristics. It then proceeded to the presentation of the various implementations of the sub-algorithm responsible for the calculation of suggestions and the integrated tools afterwards. In the end, the entire algorithm was described with examples for each of the phases.

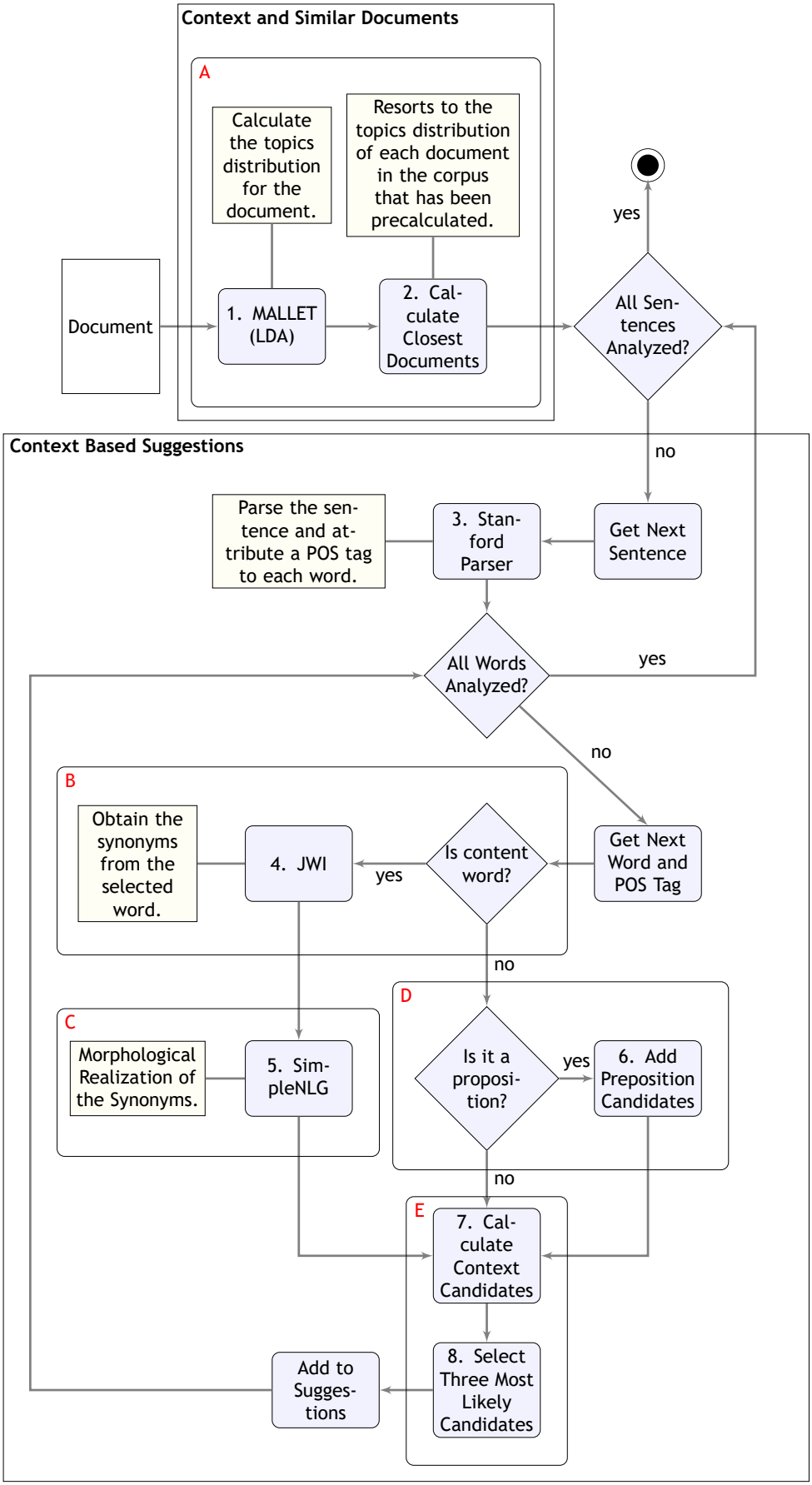

Figure 3.2: General procedure to analyze a document.

With the development of the algorithm complete, tests are required for analyzing the correctness of results and fine tunning the existing parameters, accordingly. The tests for determining the best number of topics for each of the existing subjects in the corpus were presented here and the values obtained are the ones used in the next chapter. Tests are then performed for the analysis on the best context for the calculation of suggestions.

Smoothing techniques are not used in the current version of the algorithm, even though they were introduced in Chapter 2 and *Kneser-Ney* has been effectively implemented. The reasons for not using it were the lack of information on some important details, for instance on how to select a vocabulary from a corpus, and the lack of time for performing these changes (and tests) for all tables.

Chapter 4

Tests and Prototype

4.1 Introduction

With the design and posterior implementation of the complete algorithm, testing was the next logical step. The tests were performed with the objective of analyzing the performance of the algorithm and also to fine tune its parameters. This chapter presents a battery of tests that were performed in order to evaluate the performance of the algorithm proposed. They were divided into two main subcategories: objective testing (presented in Section 4.2), where all the tests utilize a well specified metric; and subjective testing (Section 4.3), where human raters analyze the results of the application of the algorithm to a text. After the fine tunning of the algorithm, a simple text editor was developed as a proof of concept. This text editor is briefly described in Section 4.4.

4.2 Objective Testing

The quality of a document is a subjective matter and, as such, there is no standard metric for evaluating the suggestions provided by the system and how they impact the text. Given these circumstances, a battery of objective tests was created in order for the system to be tested in very specific cases.

The tests in this section are automatically created by applying changes to published scientific articles. A program developed intentionally for this purpose iterates over each document in the testing set, selects a sentence between the 10th and the 60th (preventing the selection of the initial sentences, such as title and authors, which most commonly suffer from parsing errors) and calculates its most relevant words. It then proceeds with the modification of some of them, while avoiding changes in words closer than three spaces (since trigrams are being used), to prevent changes of context.

4.2.1 Evaluation

The documents resulting from the previously described procedure are analyzed and the result is considered a success if the algorithm is capable of proposing the word that was originally used in the published document, otherwise, a failure is signaled. The metric used to evaluate the

list of possible replacement is the MRR, as suggested by [WCMC10]. MRR is calculated by:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}, \quad (4.1)$$

where Q is the list of errors and $|Q|$ its length. The $rank_i$ is the position of the correct replacement in the list of replacements for error i. When the correct position is not present in the list, $\frac{1}{rank_i}$ is considered to be zero. The values of MRR can be between 0 and 1, with 0 meaning that the answer is never on the list and with 1 meaning the answer is always at the first position of the list.

This evaluation approach is far from perfect, as it assumes that every single sentence in the selected document is already written in the best possible way. The system is penalized when it provides better solutions that differ from the original text and beneficed when it suggests the original text, thought it may not be the most correct word, thus somewhat balancing the results.

4.2.2 Discussion of Results

The whole set of results obtained during this phase of the project is included in Section B.3, structured in tables. This subsection contains solely a representative subset along with a more detailed analysis. Each table presents the results for one set of documents and a table in the database (the tables have been described in Section 3.3). The first column is the error type, with the following meanings: type 0 errors are the ones created by replacing a content word with a random synonym, representing the cases when the user fails at choosing the most adequate synonym for that specific context; type 1, created by replacing a preposition with another one, in a random manner, representing the case where users wrongly select another preposition; type 2, created by replacing a word with a predefined set of characters which form something of a placeholder, representing missing words or misspells. The second column is the number of errors created for each one of the aforementioned types. The six next columns are each of the types of suggestion:

1. The suggestions provided by the language tool;
2. The suggestions of the most likely synonyms;
3. Three types of suggestions purely based on context, namely using the three possible positions for a single word (in a trigram) when the word is at the end, the middle or at the beginning and;
4. The list of prepositions.

The values in each cell are the percentage of suggestions that could be provided using solely

Table 4.1: Results for the tests with 1030 documents from Mathematics using the `math` table, with a document threshold of 6.0.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1158	1.63 / 1.63	54.35 / 2.72	56.52 / 9.24	47.83 / 12.50	34.78 / 13.04	0.00 / 0.00	15.89	0.12
1	1946	0.00 / 0.00	0.00 / 0.00	59.57 / 11.45	53.19 / 5.51	35.80 / 4.49	65.80 / 1.45	35.46	0.30
2	1308	0.00 / 0.00	0.00 / 0.00	63.42 / 33.85	44.75 / 11.28	43.97 / 16.73	0.00 / 0.00	19.65	0.16
All	4412	0.27 / 0.27	8.84 / 0.44	59.95 / 16.18	50.40 / 7.96	37.49 / 8.66	40.14 / 0.88	25.63	0.21

that specific suggestion type and which percentage is specific to that suggestion type. The column named `Total` presents the percentage of errors which are effectively corrected by the algorithm but without taking the order of the suggestion into account, in opposition to MRR.

Table 4.1 presents the results for 1030 documents from Mathematics using the `math` table. This table contains the n-grams for the Mathematics training set with the specification of the document of origin for each n-gram. The tests used a document threshold of 6.0, resulting in the selection of an average of 90.28 documents. The results present an overall MRR of 0.21, implying a somewhat good response to the errors created during the tests. The errors of type 0 and 2 show similar levels of correction, with type 1 being the type of error where the algorithm performs best. The expectation was for this set to have the best results, since it only uses the n-grams from the most similar documents in the corpus. Nonetheless, the analysis on some of the results seems to show that the number of selected documents is too small, given that most n-grams are not represented in the table, justifying the results.

Table 4.2 presents the results for 1103 documents from Computer Science using the `csTotal` table. This is the test set with the best results, presenting an MRR of 0.36. This database table combines the n-grams from the entire Computer Science training set. These were extracted from 3305 documents which, according to the results, seem to be sufficient for capturing some of the characteristics inherent to this field. Through the analysis on the values of the total percentage and MRR, their divergence means that when the algorithm is capable of correcting the error, the correct suggestion is mostly in the first position.

The first value from the `Synonyms` column in the first data row of Table 4.2 means that, if the algorithm only used the synonyms to calculate suggestions for this error type, then it would have achieved 59.05% of the total $24,58\%$ score (resulting in 14.51%). The second value means that, if the algorithm were to provide suggestions without the `Synonyms` type of suggestions, then it would lose 6.23% of the total $24,58\%$ score (resulting in 23.05%) since the remaining suggestions provided by it are also provided by the other types.

Table 4.3 presents the results for 1031 documents from Mathematics using the `3gram` table. This is the test with the worst results, presenting an MRR of 0.17. The `3gram` database table is the one containing one million of the most frequent trigrams extracted from the COCA, including n-grams from all types of text in English. This broader representation of the English language

Table 4.2: Results for the tests with 1103 documents from Computer Science using the `csTotal` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1371	2.97 / 2.97	59.05 / 6.23	45.99 / 7.72	63.20 / 10.68	36.50 / 5.93	0.30 / 0.00	24.58	0.19
1	2063	0.68 / 0.68	0.00 / 0.00	68.26 / 4.77	70.04 / 3.06	52.09 / 2.55	70.81 / 1.70	56.96	0.48
2	1160	0.00 / 0.00	0.00 / 0.00	58.71 / 14.32	71.78 / 11.41	62.66 / 8.71	0.00 / 0.00	41.55	0.34
All	4594	0.90 / 0.90	9.98 / 1.05	62.19 / 7.57	69.31 / 6.37	52.01 / 4.61	41.78 / 1.00	43.40	0.36

Table 4.3: Results for 1031 documents from Mathematics using the `3gram` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1128	4.24 / 4.24	35.59 / 5.08	36.44 / 5.93	58.47 / 20.34	38.14 / 13.56	0.00 / 0.00	10.46	0.09
1	2045	0.15 / 0.15	0.00 / 0.00	49.07 / 8.64	48.46 / 6.48	43.06 / 7.87	64.51 / 0.77	31.69	0.28
2	1395	0.00 / 0.00	0.00 / 0.00	50.58 / 27.33	52.33 / 22.09	40.12 / 17.44	0.00 / 0.00	12.33	0.09
All	4568	0.63 / 0.63	4.47 / 0.63	47.76 / 11.72	50.42 / 11.08	41.89 / 10.34	44.56 / 0.53	20.53	0.17

and terms might be the justification for the results obtained, given that the n-grams are most likely failing to capture the specificity of the domain of Mathematics [JM00].

Table 4.4 shows the results for 1031 documents from Mathematics using the `csCut` table. This table contained n-grams extracted from documents from Computer Science, which means that the algorithm was making suggestions based on a field different from the one of the documents. Even though the results are lower than the equivalent tables from Mathematics, the difference seems to imply that the differences between the two fields are not that many.

Table 4.5 shows a summary of the results for all the objective tests performed on Computer Science and Mathematics documents. Overall, the results are consistent with the previous analyses: the tables containing the n-grams from the entire training set are the ones with the best performance (`...Total`), followed closely by those without the 1-count n-grams (`...Cut`); and the tables with n-grams from a different field perform almost as good as the equivalent table from the same field. This table presents the time it took for the algorithm to perform each test set also. If the time is considered in combination with the results, then the best method would probably be to use the `...Cut` tables, which perform almost at the level of the best, while taking less than half the time. Another point of interest is the document threshold, a small set of values was tested for the Mathematics test set with the `math` table. The results show that the performance is proportional to the document threshold, which is explained by the increase in the number of selected documents and thus increase in variety of n-grams.

Table 4.4: Results for 1031 documents from Mathematics using the `csCut` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1148	1.68 / 1.68	62.63 / 4.71	60.27 / 8.42	67.00 / 9.43	38.38 / 5.05	0.34 / 0.00	25.87	0.22
1	1935	0.00 / 0.00	0.00 / 0.00	61.95 / 6.46	65.44 / 4.00	47.38 / 2.77	65.95 / 1.33	50.39	0.42
2	1275	0.00 / 0.00	0.00 / 0.00	63.52 / 21.00	62.99 / 8.66	56.17 / 9.19	0.00 / 0.00	29.88	0.24
All	4358	0.30 / 0.30	11.25 / 0.84	62.00 / 10.16	65.15 / 6.04	47.79 / 4.65	38.95 / 0.78	37.93	0.31

Table 4.5: Summary of the results for the automatic tests.

Table	Documents Field	Number of Documents Analyzed	Documents Threshold	Average Number of Closest Documents	Duration (h)	Error Type 0 (MRR)	Error Type 1 (MRR)	Error Type 2 (MRR)	Total (MRR)
cs	cs	1104	6.0	63.82	28.26	0.12	0.33	0.21	0.24
csTotal	cs	1103	-	-	8.15	0.19	0.48	0.34	0.36
csCut	cs	1103	-	-	2.76	0.19	0.47	0.29	0.34
math	cs	1103	6.0	100.54	16.23	0.09	0.27	0.15	0.18
mathTotal	cs	1105	-	-	4.56	0.16	0.42	0.28	0.30
mathCut	cs	1105	-	-	2.19	0.15	0.40	0.27	0.29
3gram	cs	1103	-	-	1.62	0.12	0.35	0.14	0.23
cs	math	1029	6.0	65.88	24.16	0.13	0.29	0.15	0.21
csTotal	math	1031	-	-	7.42	0.20	0.42	0.25	0.31
csCut	math	1031	-	-	2.67	0.22	0.42	0.24	0.31
math	math	1030	6.0	90.28	9.46	0.12	0.30	0.16	0.21
math	math	1033	5.0	54.25	5.17	0.11	0.24	0.13	0.17
math	math	1033	4.0	8.72	2.77	0.08	0.19	0.11	0.14
math	math	1035	3.0	3.62	2.17	0.05	0.15	0.09	0.11
mathTotal	math	1030	-	-	4.67	0.23	0.45	0.28	0.34
mathCut	math	1029	-	-	2.16	0.25	0.45	0.28	0.35
3gram	math	1031	-	-	1.51	0.09	0.28	0.09	0.17

4.2.3 Failed Cases

This subsection presents some cases where the algorithm failed at proposing the correct words. The failed cases can be mostly divided into:

- parsing errors – the errors resultant from the parsing of the original pdf document affect several parts of the algorithm and of the method used for testing. The syntactic parsing performed on the sentences of the documents is one of the steps affected by these errors, resulting in the incorrect attribution of POS tags to some of the words and disrupting the correct calculation of suggestions. Another mistake induced by this type of errors happens during the creation of the tests. If the word selected as a test contains parsing errors, then the algorithm is most likely going to fail at proposing it, since its probability in the corpus should be rather small;
- conjugation problems – given that the algorithm is only capable of performing the morphological realization of a small number of words, it frequently continues to propose the base form, even when the original word was in an inflected form, resulting in poorer results;
- lack of n-grams – too many cases are being assigned with the backup probability due to the lack of n-gram counts in the database tables, resulting in a somewhat random choice of words for suggestions. One probable solution for this problem would include the implementation of a smoothing algorithm, which could most likely provide more accurate probabilities for the n-grams.

4.3 Subjective Testing

The algorithm was subjected to a smaller battery of subjective tests at a later stage of the project, further supporting the analysis performed through objective testing.

To perform these specific tests, a section of this very dissertation was selected and submitted

to the algorithm. The respective excerpt of text was written and got no exterior revisions prior to the analysis. The result of the analysis comprised the full list of suggestions, including all the candidates for each word in the document, proposed using the `csTotal` table as the source of n-grams. With the suggestions proposed from the algorithm, two new versions of this section were created: the first resulted from the replacing of each word with the best replacement proposed by the algorithm, which excluded words whose replacement list contained themselves (Dummy Version (DV)); and the second was a similar procedure, but with a more intelligent selection of replacements, where the change could be done by any word with any form of the words proposed by the algorithm (Intelligent Version (IV)).

Examples 13, 14 and 15 show the original first paragraph of that section (Original Version (OV)), the paragraph after the dummy application of changes (DV) and the paragraph after performing a thorough and intelligent choice of modifications (IV), respectively. The background color of the words that were changed between versions was highlighted. The red background color represents a change with loss of meaning, while the blue background represents a change that does not disrupt the content. The three complete versions of the section are presented in Section B.4.

Example 13. Generative probabilistic topic modeling is a group of algorithms that find topics by considering that each document in the collection is created by a process called the generative process. This process considers the existence of a latent structure, also known as hidden, that was used in the generation of the documents. The objective of this set of algorithms is then to reconstruct the structure, resorting to the observed variables which are, in most cases, the words of each document in the collection. As for the hidden structure, it is composed of (latent) variables that vary from model to model but, that generally include a probability distribution over topics, when the model considers each document a mixture of topics, representing the possibility that a document depicts more than one topic.

Example 14. Generative probabilistic topic modeling is a group of approaches which find topics by showing beyond framework document in a collection is defined by a process , the generative process. This process considers the existence of a conceptual structure, also known as hidden one that was used in the generation of class documents. The objective of the set of programs is then to reconstruct the structure of according to the observed variables there represent shown in most cases, the words of framework document in a collection. As for instance hidden structure, it is made of (latent) variables that vary from model to model however , because generally include a probability distribution over topics, if the model considers each document a mixture of topics in stand for a possibility as enum document depicts more than one topic.

Example 15. Generative probabilistic topic modeling is a series of algorithms which detect themes by assuming that each document from the collection is defined by a procedure known as the generative process. This process conceives the existence of a latent structure, also referred as hidden, that was employed in the generation of the documents. The objective of this set of algorithms is then to reconstruct the structure, resorting to the observed variables which represent , in most cases, the words from each document of the collection. As for the hidden structure, it is comprised of (latent) variables that differ between models but, that in general define a probability distribution over topics, when the model considers each document a mixture of topics, representing the possibility that a document portrays more than one topic.

The original section in Example 13 seems to reflect the use of a poor vocabulary by the author, falling in a repetition of terms or in the usage of mostly basic terms. The application of the replacements proposed by the algorithm without a proper analysis, as in Example 14, leads to a complete change of intention and subject of the text, thus resulting in an inappropriate use of the algorithm. On the other hand, the intelligent approach shows that the author is capable of deciding which words should or should not be changed, with the added bonus of being able to correctly inflect some of the words proposed by the algorithms in its base form. This form seems to help the author with a variety of synonyms, enhancing the overall quality of the text.

The tests were then expanded to the remaining paragraphs of the section and to a small group of subjects. Each test contains the three versions of a paragraph randomly ordered. Subjects with some knowledge in the area were tasked with selecting the best paragraph from each test. The results are presented in Table 4.6, with each row being the answers of a subject to each test. The tests can be described as follows:

- `Test A` contained the first paragraph, with the order being DV, OV, IV;
- `Test B` contained the second paragraph, with the order being IV, OV, DV;
- `Test C` contained the third paragraph, with the order being OV, DV, IV;
- `Test D` contained the fourth paragraph, with the order being DV, IV, OV;
- `Test E` contained the fifth paragraph, with the order being IV, DV, OV.

Table 4.6: Results for the subjective testing of the suggestions proposed by the algorithm.

Test A	Test B	Test C	Test D	Test E
3 (IV)	2 (OV)	3 (IV)	3 (OV)	1 (IV)
2 (OV)	2 (OV)	1 (OV)	3 (OV)	1 (IV)
2 (OV)	2 (OV)	1 (OV)	3 (OV)	3 (OV)
3 (IV)	2 (OV)	2 (DV)	3 (OV)	1 (IV)
3 (IV)	1 (IV)	3 (IV)	2 (IV)	1 (IV)
2 (OV)	2 (OV)	3 (IV)	3 (OV)	1 (IV)
3 (IV)	2 (OV)	2 (DV)	3 (OV)	1 (IV)
3 (IV)	2 (OV)	3 (IV)	3 (OV)	3 (OV)

The results present three tests where most people agreed on their choice: for `Test B` most subjects choose the Original Version; for `Test D` they also choose Original Version; and for `Test E` they choose the Intelligent Version. As for tests `A` and `C` the choices were somewhat balanced between the OV and the IV. `Test C` is the only one where two people choose the DV paragraph as the best. This is most likely justified by the small number of changes performed, which is a consequence of the short size of the corresponding paragraph. A direct count on the responses from the subjects shows: 21 counts for the OV; 2 counts for the DV; and 17 counts for the IV.

The results from the subjective tests show that all subjects agree with the fact that the procedure for the generation of the Dummy Version is not a good method, disrupting the flow and intention of the text. A deeper analysis of these results is performed in Chapter 5.

4.4 Proof of Concept

The resulting algorithm (Section 3.6) was integrated in a simple text editor developed as a proof of concept for the implementation of this type of suggestions in real life scenarios. It was configured with the best parameters obtained from testing, including the number of topics, the most appropriate table from the database and the document threshold. The text editor is very simple, following the software engineering process detailed in Appendix A.

The system was designed with the main objective of providing productive suggestions to the user, allowing for the improvement of the scientific content of the document. It should help the user in a preemptive manner, with minimal interaction and effort, while allowing him or her to maintain complete control over the final result. In order for the system to be usable, basic functionalities, common to most text editors and most tools nowadays, needed to be included. Some of the functionalities are the opening of existing documents and the find/search subsystem. The system implements simple versions of these functionalities, which allow for the user to open any text document and finding words or regular expressions in the text.

The editor tracks changes in the text in real-time, forming a list of sentences and annotating which ones are changed for future analysis. Providing suggestions in real-time was not possible

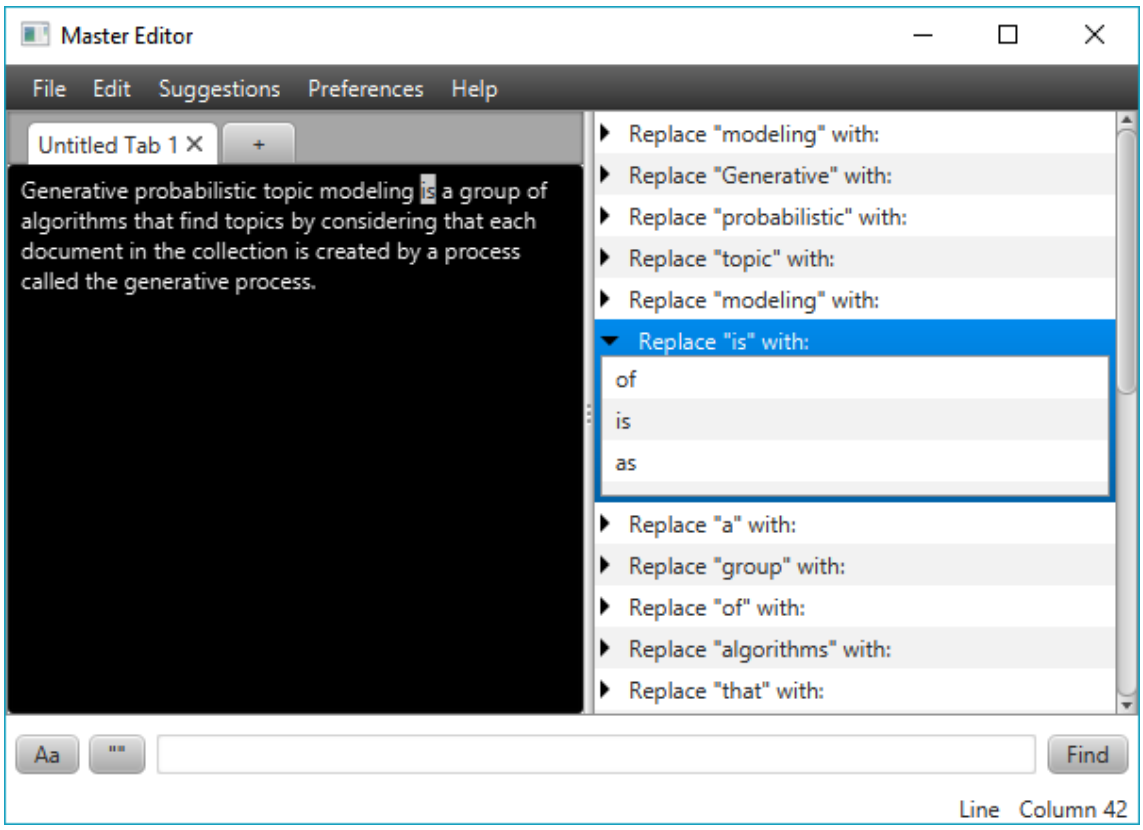

Figure 4.1: Screenshot of the text editor developed in the scope of this project.

for the current version, mostly due to the large amount of data (which, depending on the table, could reach up to 18 million entries) and the number of queries that are needed for each word in each sentence (up to ten). The solution found was to calculate suggestions in the background, after a predefined time interval or when requested by the user.

Each time a sentence is changed or a suggestion is applied, every suggestion proposed for that sentence is automatically removed, as they are no longer applicable given that the context has changed. The next analysis will once again process this new sentence and provide new suggestions according to the new context.

A screenshot of the text editor can be observed in Figure 4.1, including a representation of the suggestions provided for the sentence typed in the left side pane. The editor has two main sections, the left one is where the user introduces the text and the right one is where the suggestions are presented. By selecting a word in the suggestions section, that word is highlighted in the left section and a list of possible replacements is shown in the right section. The user can then apply a suggestion by double clicking a replacement. The word in the text will be immediately replaced.

4.5 Conclusion

This chapter presented the specification of the tests performed and the discussion of the results achieved. The test sets were of two types: objective, measuring the algorithm performance in specific cases; and subjective, resorting to humans for their opinion on the results. The chapter finalized with an overview over the text editor that was developed as a proof of concept.

An analysis on the importance of each type of suggestions in the results indicates that the context suggestions are the ones with the biggest impact in the performance of the algorithm. The second value of each cell is the percentage of correct suggestions provided exclusively by each type of suggestion. The values in the suggestions from `Language Tools`, `Synonyms` and `Prepositions` are mostly very low, under 1%, meaning that at least one of these types of suggestions could be suppressed with minimal impact in the performance of the algorithm. One reason that might justify the lack of improvements provided by the *Synonyms* suggestions is the fact that most words are still being proposed in its base form, even with the combination of `SimpleNLG`.

The fine tunning of the algorithm includes the tests performed, in the previous chapter, for the discovery of the number of topics to use (850 for both the Computer Science and Mathematics sets). The appropriate selection of the table to use, which, as shown by the results, should be the (`...Total`) table from the area of interest is another tunning feature. The last tuning option is the definition of the document threshold, which, due to limitations of time, was restricted to a small set of values, with the best performance achieved with a document threshold of 6.0, but at the cost of increased amounts of time. Perhaps with a bigger collection, a smaller value of threshold could be used to select the same amount of documents, with a better performance given that the files selected would be more specific to that context.

The next chapter presents an analysis on the results and main conclusions that resulted from the work performed under this scope.

Chapter 5

Conclusions and Future Work

This chapter presents the final remarks of this work. It includes a discussion on the objectives, an analysis of the obtained results and the identification of several different ideas for future work.

5.1 Objectives

The main objective of this work was the development of an algorithm capable of proposing suggestions for improving or correcting mistakes in a document, according to its specific context. The development of the algorithm was completed, with the automatic extraction of a representation of the context being done using topic modeling, and the matching of similar documents performed via comparison of these representations. The application of the algorithm to a text document gives rise to a list of candidate replacements for each word, effectively selecting those that are most likely to appear in the closest documents.

Another objective was the evaluation of the developed algorithm, which was performed via combined analysis on two main testing approaches. The first approach included objective tests, which were automatically generated from the test documents of the corpus. Different parts of the documents were changed in order to simulate errors that commonly occur during the writing process. Each test is considered a success if the algorithm proposes the original word as a replacement for the changes performed. The second approach resorted to the opinion of human raters towards three different versions of the same section, two of those containing suggestions proposed by the algorithm.

The last objective was the integration of the algorithm in a functional text editor as a proof of concept. In order to fulfill this objective, a simple text editor was developed with the algorithm in mind. The text editor automatically splits and analyses each sentence of the text with the algorithm, displaying the proposed suggestions to the user. The user is capable of defining the document threshold for the selection of closest documents, as well as selecting which suggestions to apply. The source code for the text editor is publicly available at [Cor16], including its Java documentation (Javadoc), as suggested in this Master's proposal.

5.2 Results and Conclusions

This Master's project targeted a problem felt by many investigators when writing scientific documents for sharing their findings with the community. The problem includes the need to ensure that the text written conveys the exact meaning intended by the authors and makes appropriate usage of the terminology specific to that area of knowledge. The proposed solution comprised the development of an algorithm, which would analyze the text from a document and calculate suggestions for improvements and corrections, based on the document itself and on related documents from the same field.

Through the gathering of a collection of documents and the integration of several NLP tools, the implemented algorithm is capable of first selecting the closest documents to the one being written and then using them as the source for the calculation of the replacements which are more likely to appear. Tests were prepared for evaluating the performance of the algorithm, both at an objective level with automatically generated errors, as well as at a subjective level through the rating of humans.

The results from the objective tests show a clear approximation of the text in the document to the published documents selected as the source. Being able to propose suggestions with an MRR of 0.35 means that the algorithm is capable of recovering over a third of the words originally contained within the documents that were automatically altered. This corroborates the fact that the algorithm is partially capturing very specific linguistic features of the field.

The results from the subjective tests provide a more confusing indication, with the choices of the subjects slightly leaning towards the original versions of the paragraphs. Perhaps the choice of removing the filter that selected the three most likely candidates from the entire candidate list was not a good idea. This choice was made for the Intelligent Version of the paragraphs in order to contain a bigger number of changes, otherwise resulting in very similar paragraphs. The large number of choices might have resulted in the confusion that these were all likely to be used at that context, thus resulting in the selection of bad replacements.

The combination of the results from both types of tests leads to the conclusion that the algorithm is capable of helping with the difficulties felt by investigators in the writing of this type of documents. Nonetheless, it requires a very thorough analysis of the candidate suggestions proposed by the algorithm, resulting in a cumbersome process. As such, perhaps a better approach would be to only check the suggestions proposed for the words in doubt, requiring far less effort while still benefiting from its use.

The proposed solution shows an emphasis on the context, both at the sentence level, where the words are analyzed according to their surrounding words, and at the document level, by resorting to the closest documents. These point towards the conclusion that the the context is

a key point on the analysis of the quality of documents, which is further strengthened by the high percentage of correct suggestions provided by the algorithm.

The implementation of the text editor with the algorithm and the tests performed have demonstrated that even though the algorithm can be successfully employed in the improvement of the scientific coherence of a document, it takes too long for it to able to produce real time suggestions. The alternative solution, found and developed, helps improve the usability of the algorithm by allowing the user to continuously work while the suggestions are being calculated on the background.

5.3 Future Work

Mainly due to the lack of time and information on key details, the integration of the Kneser-ney smoothing did not happen posterior to its implementation. As shown before, this type of techniques is very important for achieving a more realistic probability calculation for the n-grams and, consequently, expressions or sentences. Once this addition has been made, new tests should be performed and the results compared to those presented here.

It is envisioned to make the findings and tool available in the form of a web application in the future, to potentially foster its usage in the scientific community. It will be required to redesign the interface in HyperText Markup Language 5 (HTML5) and the controllers in a web friendly language. Nonetheless, it will not be required to change the information model. The application logic may eventually use the Java core classes developed in the scope of the project or, alternatively, be translated into another application as well.

Several ideas arose during the course of this project, some of which could be studied as a complement to the work herein. The first concerns the exploration of expressions and collocations, instead of resorting only to words for the analysis of a document. As described by [WMW07], word order is important and phrases contain more information as a whole than the sum of its individual parts (words). This study could begin with the implementation and analysis of TNG proposed by X. Wang *et. al.*, [WMW07], which creates topics based on expressions and words. Another tool within this realm is Senta [DGGPL00], which evaluates a text in the search for multiword expressions.

As shown by the study performed on the writing of scientific documents, the documents follow a rigorous structure, with each section describing different topics of the work. As such, each section is also characterized by differences in the writing, using different verb tenses and persons or having different specificity of its sentences, among other. The proposal of suggestions while taking the specific section in which the text is included would possibly translate into more suitable suggestions.

Another interesting option would be to resort to corpus managers. This option would include the testing of the algorithm in different corpus available in such managers, such as the SciTex [DOKLK+13], and using the corpus manager to analyze the created corpus.

Bibliography

[BCD10] D. Blei, L. Carin, and D. Dunson. Probabilistic topic models. *IEEE Signal Processing Magazine*, 27(6):55–65, Nov 2010. 17

[BDD+08] Steven Bird, Robert Dale, Bonnie J Dorr, Bryan R Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir R Radev, and Yee Fan Tan. The acl anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *LREC*, 2008. 29

[Ble12] David M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, April 2012. 14, 25

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003. 15, 16

[CG96] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics. xi, 7, 8, 9, 10, 13, 14, 24, 25

[CH01] David A. Cohn and Thomas Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 430–436. MIT Press, 2001. 18

[CK11] Ching-Fen Chang and Chih-Hua Kuo. A corpus-based approach to online materials development for writing research articles. *English for Specific Purposes*, 30(3):222 – 234, 2011. ix, 2

[COJ97] Gregory KWK Chung and Harold F O'Neil Jr. Methodological approaches to online scoring of essays. 1997. 23

[Cor16] Acácio Correia. Master editor - text editor for scientific documents, 2016. [Online; accessed on 07/10/2016]. Available from: `https://bitbucket.org/correia55/mastereditor/`. 53

[DGGPL00] Gaël Dias, Sylvie Guilloré, and José Gabriel Pereira Lopes. Benefiting from multidomain corpora to extract terminologically relevant multiword lexical units. In *9th EURALEX International Congress*, pages 339–348, Stuttgart - Germany, 2000. 55

[Dik06] Semire Dikli. An overview of automated scoring of essays. *The Journal of Technology, Learning and Assessment*, 5(1), 2006. 23

[DOKLK+13] Stefania Degaetano-Ortlieb, Hannah Kermes, Lapshinova-Koltunski, Ekaterina, and Elke Teich. *New Methods in Historical Corpus Linguistics*, chapter SciTex – A Diachronic Corpus for Analyzing the Development of Scientific Registers. Narr: Tübingen, 2013. 5, 29, 56

[Fin14] Mark Alan Finlayson. Java libraries for accessing the princeton wordnet: Comparison and evaluation. In *Proceedings of the 7th International Global WordNet Conference (GWC 2014), Tartu, Estonia*, pages 78–85. Global WordNet Association, 2014. 35

[Fou10] Apache Software Foundation. Apache PDFBox - Java PDF Library, 2010. [Online; accessed on 02/09/2016]. Available from: `http://pdfbox.apache.org/`. 30, 36

[GD10] H. Glasman-Deal. *Science Research Writing for Non-native Speakers of English*. Imperial College Press, 2010. 21

[GR09] Albert Gatt and Ehud Reiter. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, pages 90–93, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. 36

[GS04] Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004. 16, 19, 34

[GSBT05] Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. Integrating topics and syntax. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 537–544. MIT Press, 2005. 18, 37

[Hof99a] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, pages 289–296, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. 16, 18

[Hof99b] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM. 15, 16, 18

[HZW10] P. Han, Z. Zhu, and Q. Wei. An analysis of disciplinary variation in the structure of research article introductions. In *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, pages 1–4, Dec 2010. 21

[JM00] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech*

Recognition. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000. 7, 8, 9, 10, 31, 46

[KM03] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. 35

[KOTT03] J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–i182, 2003. 29

[Lib98] Cornell University Library. arXiv - electronic archive, 1998. [Online; accessed on 03/09/2016]. Available from: `http://arxiv.org`. 5

[LN13] Annie Louis and Ani Nenkova. A corpus of science journalism for analyzing writing quality. *Discourse and Dialogue*, 4(2):87–117, 2013. 22

[LS15] S. Louvigné and J. Shi. Corpus-based analysis of academic RA genre: The "Results" sub-genre. In *2015 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pages 185–189, Aug 2015. 1, 2, 21

[McC02] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit, 2002. [Online; accessed on 02/09/2016]. Available from: `http://mallet.cs.umass.edu`. 34

[Mil95] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, November 1995. 35

[Mił10] Marcin Miłkowski. Developing an open-source, rule-based proofreading tool. *Software, Practice and Experience*, 40(7):543–566, 2010. 35

[Nab03] Daniel Naber. A rule-based style and grammar checker. Diplomarbeit Technische Fakultät, Universität Bielefeld, Germany, 2003. 35

[NSH12] N. Nakmaetee, M. Sodanil, and C. Haruechaiyasak. Verb suggestion for english academic writing using wildcard query. In *Computing and Convergence Technology (ICCCT), 2012 7th International Conference on*, pages 668–672, Dec 2012. 1, 2, 24

[Nwo97] Kevin Ngozi Nwogu. The medical research paper: Structure and functions. *English for Specific Purposes*, 16(2):119 – 138, 1997. 21

[OC09] Stephen T. O'Rourke and Rafael A. Calvo. Visualizing paragraph closeness for academic writing support. *2014 IEEE 14th International Conference on Advanced Learning Technologies*, 0:688–692, 2009. 23

[Ree12] Colorado Reed. Latent dirichlet allocation: Towards a deeper understanding, 2012. 25

[Rei06] Arianne Reimerink. The Use of Verbs in Research Articles: Corpus Analysis for Scientific Writing and Translation. *New Voices in Translation Studies*, 2, 2006. 20, 21

[RKEO15] Christian Rapp, Otto Kruse, Jennifer Erlemann, and Jakob Ott. Thesis writer: A system for supporting academic writing. In *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing*, CSCW'15 Companion, pages 57–60, New York, NY, USA, 2015. ACM. ix, 1, 23

[RZGSS04] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 487–494, Arlington, Virginia, United States, 2004. AUAI Press. 18

[San90] Beatrice Santorini. Part-of-speech tagging guidelines for the penn treebank project (3rd revision). 1990. 35, 37

[SG] M. Steyvers and T. Griffiths. *Latent Semantic Analysis: A Road to Meaning*, chapter Probabilistic topic models. Laurence Erlbaum. 20

[SSDL+14] Duygu Simsek, Simon Buckingham Shum, Anna De Liddo, Rebecca Ferguson, and Ágnes Sándor. Visual analytics of academic writing. In *Proceedings of the Fourth International Conference on Learning Analytics And Knowledge*, LAK '14, pages 265–266, New York, NY, USA, 2014. ACM. 23

[Sut00] Brian Sutton. Swales's" moves" and the research paper assignment. *Teaching English in the Two Year College*, 27(4):446, 2000. 20

[TJBB05] Yee W. Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Sharing clusters among related groups: Hierarchical dirichlet processes. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1385–1392. MIT Press, 2005. 17

[UU15] Masaki Uto and Maomi Ueno. Academic writing support system using bayesian networks. In *Proceedings of the 2015 IEEE 15th International Conference on Advanced Learning Technologies*, ICALT '15, pages 385–387, Washington, DC, USA, 2015. IEEE Computer Society. 1, 24

[vdBB08] Antal van den Bosch and Toine Bogers. Efficient context-sensitive word completion for mobile devices. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '08, pages 465–470, New York, NY, USA, 2008. ACM. 2

[Vog00] Paul Vogt. Grounding language about actions: Mobile robots playing follow me games. In *SAB2000 Proceedings Supplement Book. International Society for Adaptive Behavior*. MIT Press, 2000. 7

[WCMC10] Jian-Cheng Wu, Yu-Chia Chang, Teruko Mitamura, and Jason S. Chang. Automatic collocation suggestion in academic writing. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, Short Papers*, pages 115–119, 2010. 24, 44

[WMSM09] Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1105–1112, New York, NY, USA, 2009. ACM. 19

[WMW07] Xuerui Wang, Andrew McCallum, and Xing Wei. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, ICDM '07, pages 697–702, Washington, DC, USA, 2007. IEEE Computer Society. 18, 55

[WTR+15] Denise Whitelock, Alison Twiner, John T. E. Richardson, Debora Field, and Stephen Pulman. Openessayist: A supply and demand learning analytics tool for drafting academic essays. In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, LAK '15, pages 208–212, New York, NY, USA, 2015. ACM. 24

[YFL02] Yang Yonglin, Yang Fang, and Yang Li. A research on english thesis writing. *China Central Radio and TV University Press*, pages 124–144, 2002. 29

[YXL10] H. Yang, J. Xu, and F. Liu. Lexical hedges in english abstract writing. In *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, pages 1–4, Dec 2010. 29

Appendix A

Software Engineering

A.1 Introduction

This appendix describes the software engineering process for the text editor, including the requirement analysis (Section A.2), use cases (Section A.3), activity (Section A.4) and class (Section A.5) diagrams. The text editor is briefly described in Chapter 1, though it received full attention at some point of this work, namely during the proof of concept section (Section 4.4).

A.2 Requirement Analysis

The requirement specification defines the main requirements a software must fulfill after the development is complete. This specification is used to ensure that the most important functionalities are implemented, and to reduce the costs associated with software development, since adding new functionalities/properties becomes more expensive with the advance in the development cycle.

A.2.1 Functional Requirements

The functional requirements identified for the text editor prototyped in the scope of this work were the following:

- allow the user to open, edit and save text documents;
- automatically retrieve the context from the document. This context should describe the syntactic structures and most common words that are involved in the writing of documents of the same type;
- automatically retrieve similar documents from a corpus of scientific documents, based on the context;
- allow the user to define the desired level of proximity for the similar documents;
- automatically infer the syntactic structure of the sentences;
- automatically gather synonyms or synonymous expressions from the text in the document;
- automatically suggest changes, such as word replacements, that approximate the document according to its context;

- automatically suggest corrections to grammatical errors;
- allow the user to interact with a suggestion, accepting the proposed changes;
- automatically change the text to apply a suggestion accepted by the user;
- allow the user to force the processing of the current text.

A.2.2 Non-functional Requirements

The non-functional requirements are characteristics and technicalities associated with the functional requirements that ensure certain properties of the system. These requirements are presented alongside their category.

Starting with interoperability/portability properties, the system:

- should be portable (available for all Operating Systems);
- allow users to work in different systems, as documents should be portable between Operating Systems.

User friendliness:

- should be easy to operate the main functionalities without requiring a manual;
- the interface should be as simple as possible, preventing distractions from the core work.

In terms of maintainability/extensibility:

- the system should be developed in a modular manner, allowing for the addition of new modules that introduce new functionalities;
- the source code should be open, allowing the continuous development by the community.

As for performance properties, the text editor should:

- require low specification hardware (at the level of state of the art portable computers), as a way of meeting a greater audience;
- calculate suggestions while ensuring that the processing of the input is uninterrupted, allowing the user to continue with the writing.

The response time:

- for finding the context of the document should be inferior to thirty seconds;

- for calculating and presenting the suggestions should be inferior to five minutes.

A.3 Use Cases

This section presents the use cases for the primary features provided by the proposed system. The use case diagrams represent the system seen by the perspective of the user, displaying the user interactions and the system responses. The user is the only actor of the system and it represents any person whose objective is to write or improve a scientific or academic document.

A.3.1 Open, Edit and Save Text Files

The first use case depicts the main function of any text editing tool, which is to open, edit and save text files. These use cases are represented at figure A.1 and described in detail in table A.1.

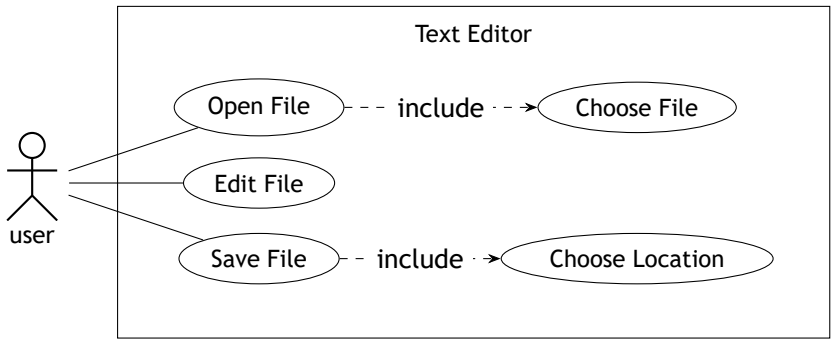

Figure A.1: Representation of the use case for opening, editing and saving files.

Table A.1: A description of the use case for opening, editing and saving files.

Actors	Description
User	The user is able to open text files by browsing through the file system and selecting the file. The text editor then tries to open and show the text from the file, allowing its edition and, when chosen, saving of the result. The document can be saved to the same file or to another one, by selecting the location and name of the file. **Note:** Opening any type of file will still result in reading the file as a text file.

A.3.2 Change Proximity Level

The next use cases concern the ability to change the desired level of proximity for similar documents, directly affecting the proposed suggestions. Its representation can be observed in figure A.2 and a more detailed description is provided in table A.2.

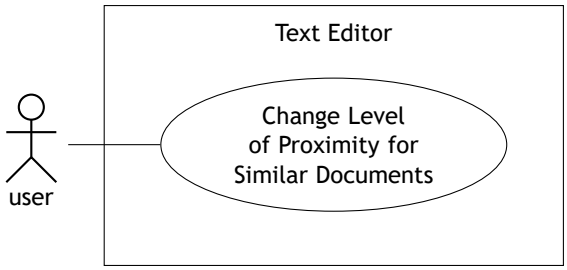

Figure A.2: Representation of the use case for changing the proximity level of similar documents.

Table A.2: Description of the use case for changing the proximity level of similar documents.

Actors	Description
User	The user can change the level of desired proximity, defining the file threshold for the acceptance of similar documents. Increasing this value represents a generalization in the area, which results in the inclusion of increasingly different documents in the suggestions calculation.

A.3.3 Interact with Suggestions

The functionalities associated with the capabilities of the user interacting with the suggestions that are provided by the editor are represented in figure A.3 and described with greater detail in table A.3.

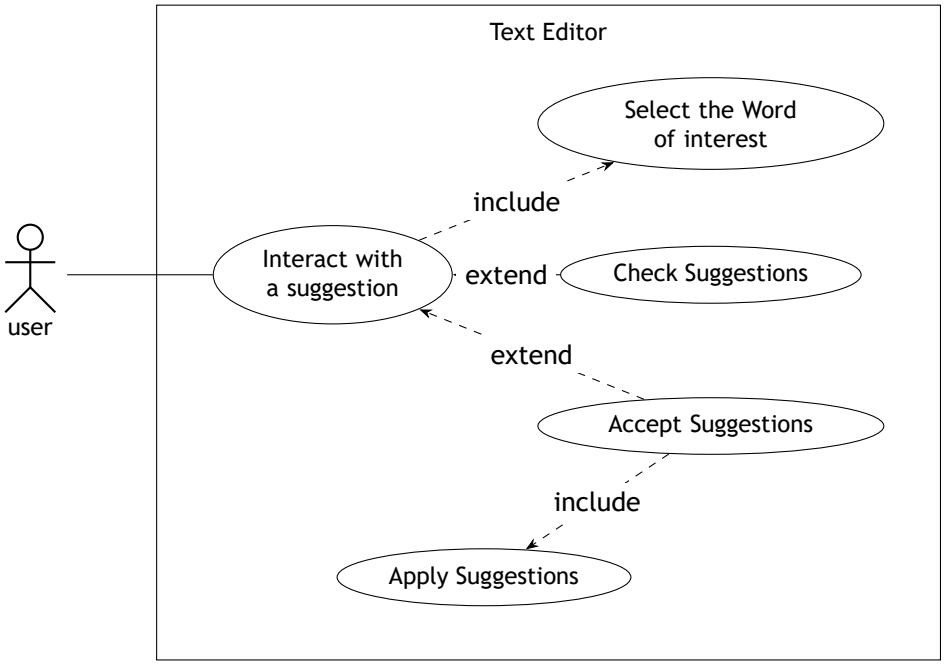

Figure A.3: The use case for the user interaction with a suggestion.

Table A.3: Description of the user interaction with a suggestion.

Actors	Description
User	After the suggestions have been calculated, the user can select a word, and check the proposed suggestions for that word. The user is then capable of analyzing each suggestion and accept it, if its his or her will. By accepting the suggestion, the system automatically makes the necessary changes. After accepting a suggestion, every other suggestion available for the same sentence is removed, avoiding the proposal of changes inappropriate for the new context.

A.3.4 Force Analysis

The user is capable of forcing the system to start a new analysis on the system, restarting all the associated processes. This action is depicted in figure A.4 and described with further detail in table A.4.

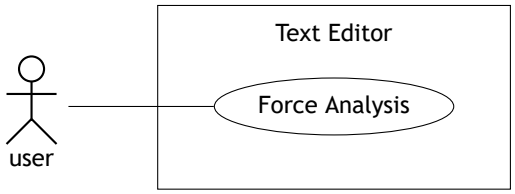

Figure A.4: A representation of the use case for forcing an analysis.

Table A.4: Description of the use case for forcing a new analysis

Actors	Description
User	The user forces a new analysis on the text, restarting all the associated processes, including the search for context and related documents, and the calculation of the suggestions.

A.4 Activity Diagrams

The activity diagrams are used to describe the flow between correlated activities. Given the simple nature of most actions performed in the text editor, this section contains only one diagram for the overall activities between the user and the system.

Table A.5 describes the steps involved in the main functions of the system, represented in Figure A.5. It combines both the treatment of the user input and the simultaneous calculation of suggestions, following a conceptual perspective.

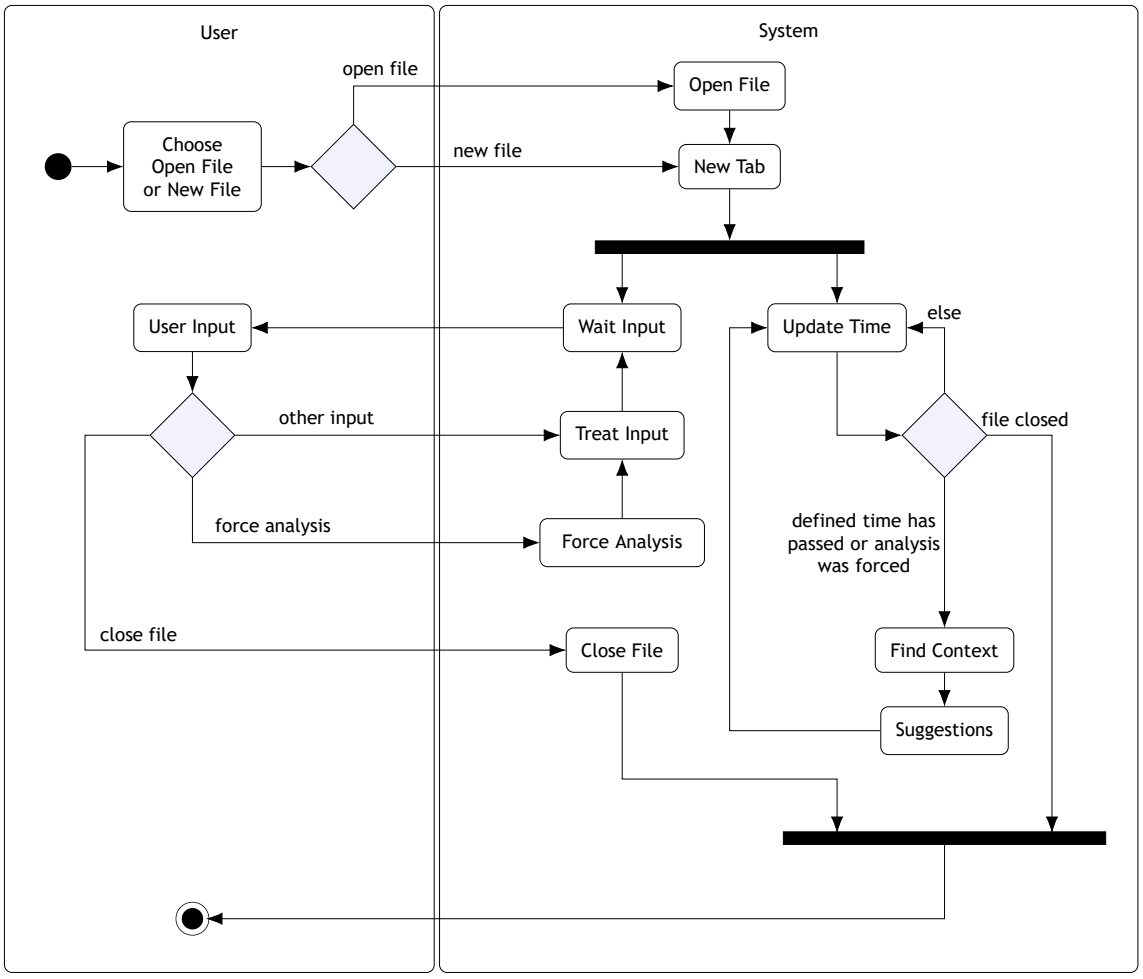

Figure A.5: Activity diagram representative of the activities responsible for the main functions provided by the system.

Table A.5: Description of the main activity diagram.

Steps	Description
1	The process starts after the user decides to either open an existing file or to create a new one.
2	The system then processes this input creating a new tab and, if the selected option was to open a file, displaying the text from the file.
3	A new thread should be created with the objective of calculating the suggestions in the background, while the main thread processes the user input.
4	The user can then introduce text, choose to force an analysis or close the file. The first two actions result in the appropriate process by the system and posterior wait for a new input to process. The latter results in the system closing the file and ending the entire process.
5	Once the defined time has passed or the user has chosen to force an analysis, the system starts to calculate the suggestions. The process begins with the finding of a context, followed by the calculation of suggestions according to that context. This process is repeated until the file is closed.

A.5 Class Diagrams

Class diagrams are used to describe the structure of the system in a static manner. They provide information about the classes available in the system, including their attributes and methods,

and how they interact with each other. Figure A.6 is the class diagram for the text editor, representing its main classes. Each class includes the name and only the most important attributes and methods, to benefit readability and the understanding of the system. A description of the purpose and importance of each class can be described as follows:

- `SuggestedReplacement` — contains a word or expression which has been proposed as a replacement for a word in the text under analyses. Each suggested replacement is classified according to the method behind the suggestion. The considered types are: Language tool, synonyms, prepositions and each of the types of context suggestions (one for each position of the trigram);
- `Suggestion` — a suggestion groups a set of suggested replacements concerning the same word in the text and contains the position of the word inside the corresponding sentence;
- `Sentence` — a sentence is represented by the text of the sentence, the position of the sentence in the text and a list of all the suggestions that concern words in that sentence. An extra variable has been added to describe whether the sentence has been updated since the last analysis of the text;
- `Document` — a document stores a list of all the sentences it contains and a context, which is defined by its closest documents;
- `FileContext` — lists the closest documents to a document;
- `EditorDocument` — is an extension to `Document`, adding variables necessary to the editor. The stack of changes that form both the undo and redo stacks are some of these variables, and the `TextArea` associated with each document is another one;
- `FXMLDocumentController` — is the main class, which is responsible for coordinating most of the work performed by every other class. It performs all the actions that are available in a regular text editor: processing input, opening, saving and closing files. Other functions include the automatic parsing of the text into sentences and keeping track of which ones have been modified, in real-time. It resorts to the `NGramDB` for performing the word prediction and interacts with both `LDA` and `CalculateSuggestions` objects for calculating the suggestions;
- `NGramDB` — serves as the interface with the database where the tables with the n-grams were stored. It implements all methods concerning the calculation of suggestions, querying the DBMS (`MySQL`) for the most likely words in a given position, the most likely expressions formed by a set of n-grams, or the probability of an n-gram or expression, among others;
- `LDA` — utilizes MALLET for the estimation of the probability distribution over the topics for the current document;

- `CalculateSuggestions` — corresponds to the class that combines the answers obtained from `NGramDB` to calculate the suggestions;
- `JWI` — is the class responsible for interacting with the lexical database WordNet, through the use of JWI. Its main function is to provide synonyms for a word, given its POS tag;
- `StanfordParser` — parses the grammatical structure of a sentence, determining the POS tag for each of the words;
- `LanguageTool` — provides an extra layer of grammatical corrections to the editor. It checks the text, based on a set of rules, providing an explanation for the problem and a set of replacements as a solution;
- `SimpleNLG` — performs the morphological realization of some of the words, by extracting information about the person, number and tense from a POS tag.

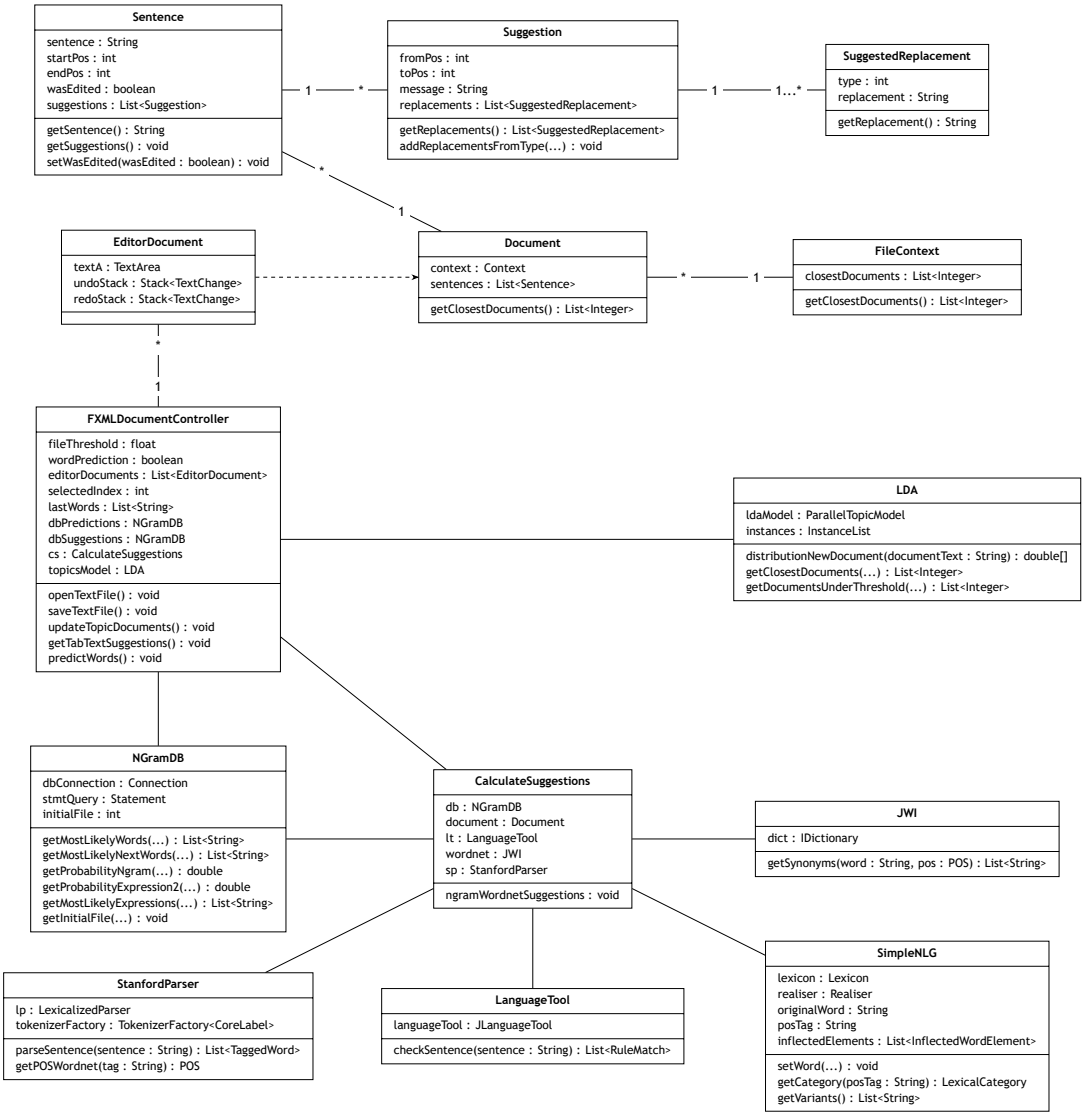

Figure A.6: Class diagram for the system from an implementation perspective.

Appendix B

Results

B.1 Introduction

This appendix contains the results obtained for the majority of the tests. They include the ones concerning choice of the number of topics (in Section B.2) and all results for the automatic tests performed to ascertain the validity of the developed algorithm, in Section B.3. Section B.4 contains the three complete versions of the text used in subjective testing.

B.2 Parameters

Figure B.1 presents charts with the probability vs. the number of topics for the fields of Mathematics and Computer Science. Topics were ranging in the interval ($[50, 1250]$). They were used to select the number of topics to be considered in the embodiment of the algorithm.

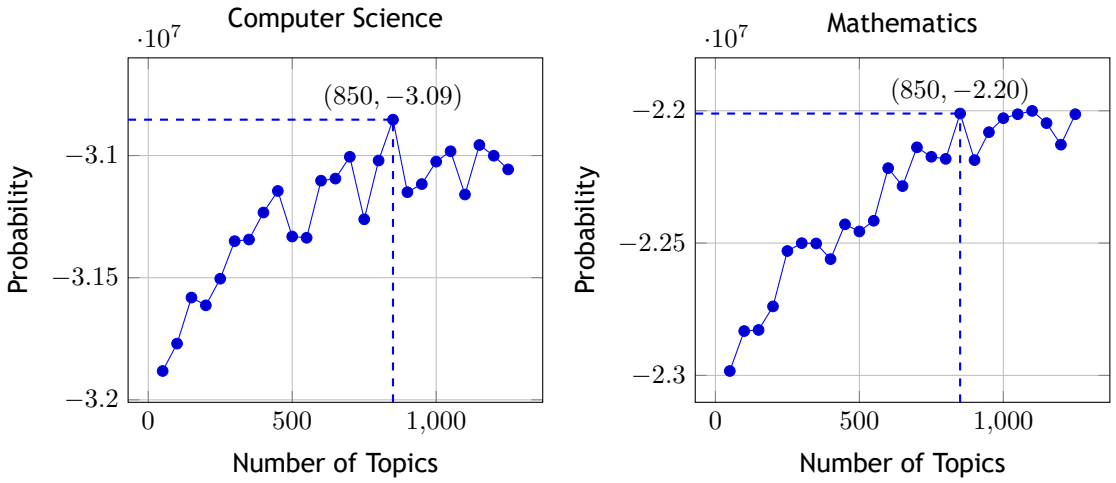

Figure B.1: Probability of the Computer Science and Mathematics validation sets for different number of topics.

B.3 Objective Tests

This section contains the results obtained during the objective testing.

Table B.1: Results for the tests with 1104 documents from Computer Science using the `cs` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1328	1.00 / 1.00	43.50 / 4.00	46.00 / 8.50	53.00 / 18.00	32.00 / 15.00	0.00 / 0.00	15.06	0.12
1	1968	0.92 / 0.92	0.00 / 0.00	58.78 / 8.06	53.37 / 3.83	39.89 / 3.96	71.99 / 1.32	38.47	0.33
2	1254	0.00 / 0.00	0.00 / 0.00	52.48 / 30.43	45.65 / 13.98	49.38 / 21.43	0.00 / 0.00	25.68	0.21
All	4550	0.70 / 0.70	6.80 / 0.63	55.20 / 13.76	51.37 / 8.60	41.05 / 10.09	42.61 / 0.78	28.11	0.24

Table B.2: Results for the tests with 1103 documents from Computer Science using the `csTotal` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1371	2.97 / 2.97	59.05 / 6.23	45.99 / 7.72	63.20 / 10.68	36.50 / 5.93	0.30 / 0.00	24.58	0.19
1	2063	0.68 / 0.68	0.00 / 0.00	68.26 / 4.77	70.04 / 3.06	52.09 / 2.55	70.81 / 1.70	56.96	0.48
2	1160	0.00 / 0.00	0.00 / 0.00	58.71 / 14.32	71.78 / 11.41	62.66 / 8.71	0.00 / 0.00	41.55	0.34
All	4594	0.90 / 0.90	9.98 / 1.05	62.19 / 7.57	69.31 / 6.37	52.01 / 4.61	41.78 / 1.00	43.40	0.36

Table B.3: Results for the tests with 1103 documents from Computer Science using the `csCut` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1429	1.77 / 1.77	55.75 / 7.08	53.69 / 5.31	58.11 / 7.37	44.25 / 11.21	0.00 / 0.00	23.72	0.19
1	2141	0.34 / 0.34	0.00 / 0.00	64.28 / 4.52	69.14 / 3.24	52.17 / 2.22	73.32 / 0.77	54.79	0.47
2	1143	0.00 / 0.00	0.00 / 0.00	62.44 / 15.96	68.54 / 10.09	56.10 / 10.09	0.00 / 0.00	37.27	0.29
All	4713	25/0.51 / 0.51	9.75 / 1.23	62.02 / 7.17	67.07 / 5.46	51.65 / 5.52	44.37 / 0.46	41.12	0.34

Table B.4: Results for the tests with 1103 documents from Computer Science using the `math` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1442	5,33 / 5,33	46,15 / 4,73	50,30 / 8,28	40,83 / 8,88	35,50 / 21,30	0,00 / 0,00	11,72	0,09
1	1993	1,13 / 1,13	0,00 / 0,00	56,96 / 8,90	55,50 / 6,31	34,95 / 3,88	68,12 / 2,27	31,01	0,27
2	1243	0,00 / 0,00	0,00 / 0,00	55,84 / 33,77	42,86 / 11,26	53,25 / 23,81	0,00 / 0,00	18,58	0,15
All	4678	1,57 / 1,57	7,66 / 0,79	55,60 / 14,44	50,20 / 7,86	39,19 / 11,30	41,36 / 1,38	21,76	0,18

Table B.5: Results for the tests with 1105 documents from Computer Science using the `mathTotal` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1412	1.74 / 1.74	55.05 / 7.67	46.34 / 8.36	57.14 / 10.10	45.30 / 10.10	0.00 / 0.00	20.33	0.16
1	1924	0.52 / 0.52	0.10 / 0.00	59.12 / 5.97	64.78 / 3.98	46.96 / 2.20	69.50 / 2.20	49.58	0.42
2	1238	0.00 / 0.00	0.00 / 0.00	53.76 / 18.91	63.78 / 12.76	53.08 / 14.12	0.00 / 0.00	35.46	0.28
All	4574	0.60 / 0.60	9.46 / 1.31	55.54 / 9.76	63.21 / 7.32	48.27 / 6.67	39.46 / 1.25	36.73	0.30

Table B.6: Results for the tests with 1103 documents from Computer Science using the `mathCut` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1363	2.31 / 2.31	50.00 / 4.62	49.23 / 7.31	56.92 / 10.00	50.38 / 12.69	0.00 / 0.00	19.08	0.15
1	2093	0.30 / 0.30	0.00 / 0.00	55.26 / 6.78	62.96 / 4.86	43.32 / 2.94	70.65 / 1.42	47.20	0.40
2	1226	0.00 / 0.00	0.00 / 0.00	53.05 / 16.75	61.68 / 10.15	59.64 / 17.01	0.00 / 0.00	32.14	0.27
All	4682	0.54 / 0.54	7.91 / 0.73	53.77 / 9.25	61.69 / 6.94	48.35 / 7.85	42.50 / 0.85	35.07	0.29

Table B.7: Results for the tests with 1103 documents from Computer Science using the `3gram` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1426	2.37 / 2.37	31.75 / 3.79	30.81 / 4.74	55.92 / 21.33	44.55 / 20.85	0.00 / 0.00	14.80	0.12
1	2021	0.72 / 0.72	0.00 / 0.00	51.38 / 7.19	46.95 / 5.15	44.19 / 6.11	71.50 / 0.36	41.32	0.35
2	1175	0.00 / 0.00	0.00 / 0.00	49.06 / 27.83	43.87 / 16.04	47.17 / 27.36	0.00 / 0.00	18.04	0.14
All	4622	0.87 / 0.87	5.32 / 0.63	47.53 / 10.25	47.93 / 9.69	44.75 / 12.16	47.45 / 0.23	27.22	0.23

Table B.8: Results for the tests with 1029 documents from Mathematics using the `cs` table. with a document threshold of 6.0.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1116	1.80 / 1.80	63.47 / 6.59	55.09 / 7.78	55.69 / 8.98	34.13 / 10.18	0.00 / 0.00	14.96	0.13
1	2008	0.00 / 0.00	0.00 / 0.00	57.99 / 10.79	52.66 / 6.62	37.12 / 4.03	65.04 / 1.73	34.61	0.29
2	1240	0.00 / 0.00	0.00 / 0.00	56.30 / 33.61	43.70 / 11.34	46.22 / 19.33	0.00 / 0.00	19.19	0.15
All	4364	0.27 / 0.27	9.64 / 1.00	57.18 / 15.27	51.18 / 8.00	38.64 / 8.27	41.09 / 1.09	25.21	0.21

Table B.9: Results for the tests with 1031 documents from Mathematics using the `csTotal` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1107	0.72 / 0.72	69.20 / 8.70	56.52 / 6.88	66.67 / 4.35	49.28 / 5.07	0.00 / 0.00	24.93	0.20
1	1990	0.00 / 0.00	0.00 / 0.00	64.08 / 5.43	67.91 / 3.92	50.00 / 2.82	67.51 / 1.91	49.95	0.42
2	1329	0.00 / 0.00	0.00 / 0.00	61.61 / 14.43	73.11 / 9.78	61.37 / 9.54	0.00 / 0.00	30.78	0.25
All	4426	0.11 / 0.11	11.37 / 1.42	62.23 / 7.86	68.96 / 5.41	52.65 / 4.82	39.96 / 1.13	37.93	0.31

Table B.10: Results for the tests with 1031 documents from Mathematics using the `csCut` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1148	1.68 / 1.68	62.63 / 4.71	60.27 / 8.42	67.00 / 9.43	38.38 / 5.05	0.34 / 0.00	25.87	0.22
1	1935	0.00 / 0.00	0.00 / 0.00	61.95 / 6.46	65.44 / 4.00	47.38 / 2.77	65.95 / 1.33	50.39	0.42
2	1275	0.00 / 0.00	0.00 / 0.00	63.52 / 21.00	62.99 / 8.66	56.17 / 9.19	0.00 / 0.00	29.88	0.24
All	4358	0.30 / 0.30	11.25 / 0.84	62.00 / 10.16	65.15 / 6.04	47.79 / 4.65	38.95 / 0.78	37.93	0.31

Table B.11: Results for the tests with 1030 documents from Mathematics using the `math` table. with a document threshold of 6.0.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1158	1.63 / 1.63	54.35 / 2.72	56.52 / 9.24	47.83 / 12.50	34.78 / 13.04	0.00 / 0.00	15.89	0.12
1	1946	0.00 / 0.00	0.00 / 0.00	59.57 / 11.45	53.19 / 5.51	35.80 / 4.49	65.80 / 1.45	35.46	0.30
2	1308	0.00 / 0.00	0.00 / 0.00	63.42 / 33.85	44.75 / 11.28	43.97 / 16.73	0.00 / 0.00	19.65	0.16
All	4412	0.27 / 0.27	8.84 / 0.44	59.95 / 16.18	50.40 / 7.96	37.49 / 8.66	40.14 / 0.88	25.63	0.21

Table B.12: Results for the tests with 1033 documents from Mathematics using the `math` table. with a document threshold of 5.0.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1171	2,41 / 2,41	46,39 / 3,01	54,82 / 12,65	46,39 / 12,05	29,52 / 13,25	0,60 / 0,00	14,18	0,11
1	1969	0,36 / 0,36	0,00 / 0,00	54,22 / 12,03	51,35 / 6,64	38,78 / 3,95	63,91 / 0,72	28,29	0,24
2	1269	0,00 / 0,00	0,00 / 0,00	54,87 / 35,90	43,59 / 10,26	50,26 / 20,00	0,00 / 0,00	15,37	0,13
All	4409	0,65 / 0,65	8,39 / 0,54	54,47 / 17,21	48,80 / 8,39	39,54 / 9,04	38,89 / 0,44	20,82	0,17

Table B.13: Results for the tests with 1033 documents from Mathematics using the `math` table. with a document threshold of 4.0.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1098	1,92 / 1,92	54,81 / 4,81	60,58 / 12,50	46,15 / 11,54	25,96 / 9,62	0,00 / 0,00	9,47	0,08
1	1965	0,00 / 0,00	0,00 / 0,00	52,76 / 11,04	49,01 / 6,84	34,22 / 7,51	66,45 / 1,99	23,05	0,19
2	1226	0,00 / 0,00	0,00 / 0,00	47,17 / 33,33	36,48 / 10,69	50,94 / 26,42	0,00 / 0,00	12,97	0,11
All	4289	0,28 / 0,28	7,96 / 0,70	52,65 / 16,20	45,81 / 8,38	36,73 / 12,01	42,04 / 1,26	16,69	0,14

Table B.14: Results for the tests with 1035 documents from Mathematics using the `math` table. with a document threshold of 3.0.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1134	0,00 / 0,00	52,86 / 2,86	60,00 / 8,57	44,29 / 14,29	31,43 / 15,71	0,00 / 0,00	6,17	0,05
1	1959	0,58 / 0,58	0,00 / 0,00	58,21 / 16,43	49,57 / 7,20	26,22 / 3,46	61,96 / 2,59	17,71	0,15
2	1322	0,00 / 0,00	0,00 / 0,00	51,08 / 37,41	29,50 / 6,47	52,52 / 31,65	0,00 / 0,00	10,51	0,09
All	4415	0,36 / 0,36	6,65 / 0,36	56,65 / 20,68	43,88 / 7,91	33,45 / 12,05	38,67 / 1,62	12,59	0,11

Table B.15: Results for the tests with 1030 documents from Mathematics using the `mathTotal` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1112	0.96 / 0.96	63.38 / 3.50	57.64 / 6.69	73.89 / 7.32	50.00 / 4.46	0.00 / 0.00	27.74	0.23
1	1985	0.00 / 0.00	0.00 / 0.00	63.20 / 5.20	73.23 / 3.22	50.33 / 1.42	65.66 / 2.74	53.25	0.45
2	1280	0.00 / 0.00	0.00 / 0.00	61.47 / 11.47	75.92 / 10.55	64.45 / 6.88	0.00 / 0.00	34.06	0.28
All	4397	0.16 / 0.16	11.01 / 0.60	61.81 / 6.97	73.99 / 5.70	53.68 / 3.26	38.40 / 1.60	41.10	0.34

Table B.16: Results for the tests with 1029 documents from Mathematics using the `mathCut` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1106	1.22 / 1.22	60.86 / 4.89	58.72 / 7.65	69.11 / 8.26	42.51 / 3.98	0.00 / 0.00	29.57	0.25
1	1963	0.00 / 0.00	0.00 / 0.00	63.52 / 6.06	68.82 / 3.75	48.22 / 2.98	65.26 / 2.02	52.93	0.45
2	1395	0.00 / 0.00	0.00 / 0.00	65.08 / 16.70	67.90 / 8.89	58.35 / 10.63	0.00 / 0.00	33.05	0.28
All	4464	0.21 / 0.21	10.89 / 0.87	63.05 / 9.04	68.63 / 5.85	49.75 / 5.09	37.11 / 1.14	40.93	0.35

B.4 Subjective Tests

This section contains the three complete versions of the text selected for the subjective testing. It resorts to the same scheme of colors described in Section 4.3, where a blue background represents a word that was altered and a red background a word that was altered with a change in the meaning. Examples 16, 17, 18 depict the original section, the section after dummy application of suggestions and the section with a thorough choice of suggestions applied, respectively.

Example 16. Generative probabilistic topic modeling is a group of algorithms that find topics by considering that each document in the collection is created by a process called the generative process. This process considers the existence of a latent structure, also known as hidden, that was used in the generation of the documents. The objective of this set of algorithms is then to reconstruct the structure, resorting to the observed variables which are, in most cases, the words of each document in the collection. As for the hidden structure, it is composed of (latent) variables that vary from model to model but, that generally include a probability distribution over topics, when the model considers each document a mixture of topics, representing the possibility that a document depicts more than one topic.

The generative model associated with PLSI is called the aspect model [Hof99a, Hof99b] (represented in Figure 2.1). The generation of each word in a document, according to the asymmetric formulation of this model, starts with the selection of a document d, with the index I d in the collection, with a predefined probability inside the collection. Then, for

Table B.17: Results for the tests with 1031 documents from Mathematics using the `3gram` table.

Error Type	Number of Errors	Language Tool (% / %)	Synonyms (% / %)	After (% /%)	Between (% /%)	Before (% /%)	Prepositions (% / %)	Total (%)	MRR
0	1128	4.24 / 4.24	35.59 / 5.08	36.44 / 5.93	58.47 / 20.34	38.14 / 13.56	0.00 / 0.00	10.46	0.09
1	2045	0.15 / 0.15	0.00 / 0.00	49.07 / 8.64	48.46 / 6.48	43.06 / 7.87	64.51 / 0.77	31.69	0.28
2	1395	0.00 / 0.00	0.00 / 0.00	50.58 / 27.33	52.33 / 22.09	40.12 / 17.44	0.00 / 0.00	12.33	0.09
All	4568	0.63 / 0.63	4.47 / 0.63	47.76 / 11.72	50.42 / 11.08	41.89 / 10.34	44.56 / 0.53	20.53	0.17

each word token, a latent class Z d,n is chosen from the probability distribution of the document over the latent classes. And, according to the probability distribution over the words in that class, a word W d,n is finally selected. This process is repeated for each of the documents in the collection.

As represented in Figure 2.1, the fact that I d is an observed variable means that the model considers only documents in the analyzed collection, and thus is unable to determine the topics for a new document afterwards.

This process is based on three main assumptions: each word is considered independent from the others when conditioned on the topic assignment, describing each document as a bag-of-words, in which the order of the words is ignored; considers that words conditioned on the topic assignment are independent of the document in which they insert themselves; the number of existing topics is considered known and fixed [BNJ03]. All of these assumptions simplify the process of recovering the latent structure, otherwise infeasible.

The LDA (Figure 2.2) considers a generative process with complementary parameters to those presented in the PLSI process with the addition of two new parameters: α and β. The α serves as a configuration parameter for the Dirichlet distribution that determines the distribution of topics for each document. On one hand, a small value of α is responsible for promoting distributions that have few topics with high probability. On the other hand, a high value of α promotes a high number of topics with identical probabilities. Similarly, the β is a configuration parameter for the Dirichlet distribution over words. A small value of β means each topic will describe few words in high probability, while a high value would describe a big number of words with comparable probabilities.

Example 17. Generative probabilistic topic modeling is a group of approaches which find topics by showing beyond framework document in a collection is defined by a process , the generative process. This process considers the existence of a conceptual structure, also known as hidden one that was used in the generation of class documents. The objective of the set of programs is then to reconstruct the structure of according to the observed variables there represent shown in most cases, the words of framework document in a collection. As for instance hidden structure, it is made of (latent) variables that vary from model to model however , because generally include a probability distribution over topics, if the model considers each document a mixture of topics in stand for a possibility as enum document depicts more than one topic.

The generative model link up with a is called the aspect model [Hof99a, Hof99b] (represented in Figure 2.1). The generation of each word in a document, according to the

asymmetric formulation of this model it starts with the selection of a document k, with the index l used in a collection, with a proper probability inside the collection. Then, for each word appear, a modified class Z l:n is decided from the probability distribution of the document over the random classes. This, according to the probability distribution over the words in the class of other word W d:n is finally decided. This process is repeated for each of the documents in a collection.

As represented in Figure 2.1, the fact that the d is an observed variable means that the model considers only documents in the same collection, and thus is unable to determine the topics for a new document afterwards.

This process is based on the main assumptions of each word is reckon independent from the text in based on the topic assignment, with each document as appear bag-of-words, in which the order of the words is snub; considers as words depending on the topic assignment exist independent of the document in that one insert data and the number of live topics is reckon spotted and level [BNJ03]. Most of these assumptions simplify the process of generating the conceptual structure, otherwise infeasible.

The joe (Figure 2.2) considers a generative process with complementary parameters than methods exhibit in the ldc's process with the addition of the new parameters; α and β. The α serves as a configuration parameter for the same distribution that determines the distribution of topics for each document. On one hand, a small value of α is responsible for providing distributions that have few topics with high probability. On the other hand, a high value of α promotes a high number of topics with identical probabilities. Similarly, there β is a configuration parameter for the resulting distribution over words. A small value of β means each topic will describe few words in high probability, with a high value would describe a big number of words with comparable probabilities.

Example 18. Generative probabilistic topic modeling is a series of algorithms which detect themes by assuming that each document from the collection is defined by a procedure known as the generative process. This process conceives the existence of a latent structure, also referred as hidden, that was employed in the generation of the documents. The objective of this set of algorithms is then to reconstruct the structure, resorting to the observed variables which represent, in most cases, the words from each document of the collection. As for the hidden structure, it is comprised of (latent) variables that differ between models but, that in general define a probability distribution over topics, when the model considers each document a mixture of topics, representing the possibility that a document portrays more than one topic.

The generative model associated with PLSI is known as the aspect model [Hof99a, Hof99b] (represented in Figure 2.1). The generation of each word in a document, consorting to the asymmetric formulation of this model, begins with the selection of a document d, with the index I d from the collection, with a predefined probability across the collection. Then, for each word token, a latent class Z d,n is chosen from the probability distribution of the document over the latent classes. And, according to the probability distribution over the words in that class, a word W d,n is finally selected. This procedure is replicated for each of the documents in the collection.

As represented in Figure 2.1, the fact that I d is an observed variable means that the model considers only documents in the analyzed collection, and hence is unable to estimate the topics for a new document afterwards.

This process is grounded on three main assumptions: each word is considered independent from the others when conditioned on the topic assignment, describing each document as a bag-of-words, in which the order of the words is disregarded; considers that words conditioned on the topic assignment are independent of the document in which they insert themselves; the number of existing topics is reckon known and fixed [BNJ03]. All of the assumptions simplify the process of recovering the latent structure, otherwise infeasible.

The LDA (Figure 2.2) considers a generative process with complementary parameters to those exhibited in the PLSI process with the addition of two new parameters: α and β. The α serves as a configuration parameter for the Dirichlet distribution which regulates the distribution of topics for each document. On one hand, a small value of α is responsible for promoting distributions that feature few topics with high probability. On the other hand, a high value of α promotes a high number of topics with identical probabilities. Similarly, the β is a configuration parameter for the Dirichlet distribution over words. A small value of β means each topic will depict few words with high probability, while a high value would describe a large number of words with comparable probabilities.