CONTEXT-AWARE SEMANTIC TEXT MINING AND REPRESENTATION LEARNING FOR TEXT DISAMBIGUATION AND ONLINE HARASSMENT CLASSIFICATION

by

Mozhgan Saeidi

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
December 2023

Dedication

To the love of my life, my loving husband, Kaveh. I would not have been standing where I am now if it were not because of your unconditional love, support, encouragement, understanding, and kindness.

فکر هر کس به قدر همت اوست. -حافظ

همت بلند دار که مردان روزگار از همت بلند بجایی رسیده اند. -سعدی

”Everyone’s thought is to the limit of their ambition” -Hafez & Sa’di, Persian poets.

Contents

List of Tables . . . . . **vi**

List of Figures . . . . . **xi**

Abstract . . . . . **xiv**

Acknowledgements . . . . . **xv**

Chapter 1 Introduction **1**

1.1 Text Ambiguity 2

1.2 Text Classification and Categorization 3

1.3 Representation Learning Development 4
1.3.1 Text Disambiguation Solutions 5
1.3.2 Text Classification Solutions 5

1.4 Goals and Contributions 6

1.5 Thesis Organization 7

Chapter 2 Contextual Knowledge-Base Representation Learning in Text Disambiguation **9**

2.1 Introduction 10

2.2 Related Work 11
2.2.1 Semantic Similarity 11
2.2.2 Text Representation Learning 12

2.3 Word Sense Disambiguation 15
2.3.1 External Knowledge Sources 16
2.3.2 Context Representation 16
2.3.3 Selection of the Classifier 16

2.4 Error Analysis Framework 18
2.4.1 WSD Experimental Setup 19
2.4.2 Error Analysis by Part-of-Speech 20
2.4.3 Results and Conclusion Based on Analyzing the Errors 30

2.5 New Proposed Representation Learning 30
2.5.1 C-KASE 31
2.5.2 Experimental Setup 33

2.6 Results . . . 37
2.6.1 Discussion . . . 41

2.7 Contextual Representation Learning in Biomedical Word Sense Disambiguation . . . 42
2.7.1 BioCBERT . . . 42
2.7.2 Experimental Setup . . . 45
2.7.3 Results . . . 45

2.8 Conclusion . . . 46

Chapter 3 Disambiguation in Wikification . . . 48

3.1 Introduction . . . 49

3.2 Related Work . . . 52
3.2.1 Formal Definition of Wikification Problem . . . 56

3.3 Methodology . . . 56
3.3.1 Notation and Definitions . . . 58
3.3.2 Coherence Measure . . . 61
3.3.3 OWCW: Overlapping-Windows Concept Wikifier Algorithm . . . 62
3.3.4 CACW: Context-Aware Concept Wikifier . . . 66

3.4 Evaluation . . . 67
3.4.1 Evaluation Datasets . . . 69
3.4.2 CACW Configuration and Parameters of Test Evaluation . . . 70
3.4.3 Baseline Wikifiers . . . 71
3.4.4 Evaluation Measure . . . 71

3.5 Results . . . 73
3.5.1 Discussion and Ablation Study . . . 79

3.6 Conclusion . . . 81

Chapter 4 Classical Machine Learning Models for Categorizing Online Harassment on Twitter . . . 82

4.1 Introduction . . . 83

4.2 Related Work . . . 84

4.3 Methodology . . . 85
4.3.1 Data . . . 85
4.3.2 Algorithms . . . 86
4.3.3 Feature Extraction . . . 87

4.4 Experiments . . . 88
4.4.1 Pre-processing . . . 88
4.4.2 Task A – Binary Classification . . . 89
4.4.3 Task B – Multi-Class Classification . . . 91

4.5 Results and Discussion . . . 94
4.5.1 Classification with TF-IDF vectors . . . 94
4.5.2 Classification with Word Embeddings . . . 96

4.6 Conclusion and Future Works . . . 98

Chapter 5 Graph Convolutional Network for Categorizing Online Harassment on Twitter . . . 101

5.1 Introduction . . . 102

5.2 Background . . . 103

5.3 Tweet Categorization . . . 104

5.4 Experiments . . . 109
5.4.1 Classical Machine Learning Algorithms . . . 109
5.4.2 Baselines . . . 109
5.4.3 Data . . . 110
5.4.4 Settings . . . 110

5.5 Results and Discussion . . . 112

5.6 Conclusion . . . 114

Chapter 6 Conclusions and Future Research . . . 117

Bibliography . . . 121

Appendix A Environment Requirements for Reproducing Results and Implementations . . . 142

A.1 Text Disambiguation . . . 142

A.2 Text Classification . . . 144

Appendix B Brief information on BabelNet . . . 146

Appendix C NASARI Vectors and Weighted Overlap Measure . . . 148

Appendix D Nomenclature **151**

List of Tables

2.1 The statistics of the dataset. 19

2.2 The accuracy of 1-NN WSD evaluation framework on the unified dataset, using the four recent contextual embedding approaches as baselines. 20

2.3 The accuracy of the 1-NN WSD of the four considered embeddings as baselines. The dataset in this experiment is a concatenation of all five datasets, which is split by Part-of-Speech tags. 20

2.4 Error frequency analysis of the 1-NN WSD evaluation framework with ARES representations on the dataset, separated by type. Freq shows the frequency of the number of each word type in the dataset, ER is the Error Rate of each model on the word types, and Mis-D shows the percentage of the words that are mis-disambiguated in each word type. 24

2.5 Following the presented result in Table 2.4, here are the statistics of the number of Disambiguated (#D) words either Correctly (C) or Incorrectly (I). The lower row under the same column shows how many of those incorrect disambiguations got corrected after using the deep learning-based POS tagger and fixing the POS tagger errors by assigning the correct type of noun. 25

2.6 The accuracy of 1-NN WSD evaluation framework on the unified dataset, using our model and comparing it with the winner of the baselines, ARES. The result of ARES is the same last row of Table 2.2, above. C-KASE is our new proposed contextual embedding model, introduced in this thesis (Section 2.5). The higher accuracies are highlighted. 35

2.7 The accuracy of the 1-NN WSD of our model, C-KASE, and ARES, which is the best one between the baselines, are taken from the last row of Table 2.3. C-KASE is our new proposed contextual embedding model, introduced in this thesis, Section 2.5. The dataset in this experiment is a concatenation of all the five datasets, which is split by Part-of-Speech tags. The higher accuracies are highlighted. 35

2.8 The accuracy of 1-NN WSD evaluation framework on the unified dataset, using our C-KASE representation, introduced in Section 2.5, and C-KASE'. The C-KASE' is the modified version of C-KASE that compares the representation of senses (extracted from the Wikipedia page of the sense) with a representation of input text. The higher accuracies in each word type are highlighted in bold. 36

2.9 The accuracy of the 1-NN WSD using the two versions of C-KASE. The C-KASE is our method introduced in Section 2.5, and C-KASE' is the modified version. This experiment indicates the effectiveness of considering contextual information from both the knowledge base and the input text during disambiguation. The higher accuracies in each word type are highlighted in bold. 37

2.10 The accuracy results of 1-NN WSD evaluation on the MSH WSD dataset, using the most recent pre-trained embedding approaches on biomedical text, and compare those embeddings with our new proposed embedding, presented in the last row. 45

3.1 The definition of True/False, Positive/Negative concerning a given sense, i-th candidate sense (c_i), from Wikipedia. The source of this table is [185]. 72

3.2 Precision, Recall, and F1-measure on the Wikification task of unsupervised knowledge-based approaches of GLOW, Wikipedia Miner, TAGME, Wikisim, and RedW in comparison with our proposed algorithms OWCW and CACW. 74

3.3 Comparing disambiguation results of baseline Wikisim approach (non-overlapping window approach) with the proposed approach (overlapping windows approach): Micro Averaged Precision ($\hat{\pi}^{\mu}$) and Macro Averaged Precision ($\hat{\pi}^{M}$), across different candidate numbers and datasets. The precision of the proposed approach (OWCW) is higher than baseline (Wikisim) across different window sizes (WS) and different datasets. The last row of T shows the run-time. 77

3.4 Run-time performance of RedW, Wikisim, and CACW on AQUAINT dataset as the short corpus, and Wiki5000 as the long corpus, reported in seconds (s). 77

3.5 Calculating the text's coherence after disambiguation using the baseline (Wikisim approach) and our two proposed models, CACW and OWCW. We use the coherence metric introduced in Eq. 3.9. 79

3.6 Ablation Study using MSNBC Dataset. All experiments use the key-entity selection algorithm CACW and are trained with the same epochs. At each experiment, we fix all but one of the variables to evaluate its effect on the model's accuracy. The averaged accuracies are reported. 80

4.1 Dataset Statistics. The detailed number of tweet posts in each harassment type of our dataset is shown in this table. 85

4.2 Acronyms and HTML tags assigned as stop words in the tweets dataset pre-processing. 88

4.3 The 45 most relevant words in the dataset after stemming with Snow-Ball stemmer. The words in the table are the direct results of the stemmer. 89

4.4 Accuracy values for the supervised methods on the validation set using features extracted by TF-IDF scores. The number of different features (most relevant terms according to TF-IDF scores) varied from 20 to 50, and each one is represented in one column. The highest results of each classifier on each feature are highlighted in bold. . . . 90

4.5 Accuracy values on the three classes of harassment over the validation set using 45 features extracted by TF-IDF scores. The top accurate classifiers for each harassment type are highlighted in bold. 93

4.6 Accuracy values for the Word2Vec embeddings on the validation set. The top accurate classifiers for each harassment type are highlighted in bold. 94

4.7 Accuracy values for the TF-IDF vectors. The first column indicates the results of each model on the binary classification task, harassment detection. The next three columns show the results of the models in the multi-class classification task. The last column shows the average accuracy of the classifiers. The top accurate classifiers for each harassment type are highlighted in bold. 95

4.8 Macro F1 values for the TF-IDF vectors on each class. The first column indicates the results of each model on the binary classification task, harassment detection. The next three columns show the results of the models in the multi-class classification task. The last column shows the average of the F1 value of the classifiers. The higher accuracies in each harassment type are in bold. 96

4.9 Accuracy values for Word2Vec Embeddings. The first column indicates the results of each model on the binary classification task, harassment detection. The next three columns show the results of the models in the multi-class classification task. The last column shows the average accuracy of the classifiers. The higher accuracies in each harassment type are in bold. 97

4.10 Macro F1 values when using Word2Vec Embeddings. The first column indicates the results of each model on the binary classification task, harassment detection. The next three columns show the results of the models in the multi-class classification task. The last column shows the average F1 score of the classifiers. The highest accuracy is in bold. 97

4.11 Average accuracy values for detecting harassment content (using scores from Table 4.7), compared to the results in the literature for detecting harassment, using the same tweet dataset introduced in this chapter. The highest accuracy is in bold. 98

5.1 Test Accuracy on both categorization tasks. We ran all models 10 times and reported the mean $\pm$ standard deviation. GCN significantly outperforms other models on this Tweet dataset, based on student t-test ($p<0.05$). 116

5.2 Test Accuracy of the model when the data is complete (GCN), in comparison with when the size of data is halved (GCN*). Here, we used SBERT embeddings which is the winner of the previous experiments. 116

List of Figures

2.1 The procedure of analyzing the errors of the contextual word embedding models in the task of WSD and evaluating the effect of different POS taggers in these errors. These numbers are shown in Table 2.5 for nouns. For the other word types, please look at Fig 2.2, which shows the number of words that have been considered incorrectly as other word types. 23

2.2 The confusion matrix for the word types in the 1-NN disambiguation task, using ARES embeddings. 26

2.3 Visualizing the positions of word embeddings and sentence embedding. 29

2.4 Demonstration of the C-KASE representation and its three components. Component 1) Collecting all Wikipedia pages for ambiguous words and possible meanings (senses), Component 2) Using hypernymy and hyponymy relations to extract all synsets for ambiguous words from Babelnet, Component 3) Concatenating (word, sense) representation for all senses from the second component with (Document's paragraph, Wikipedia's paragraph) representation from the first component as context. 34

2.5 The accuracy comparison of the C-KASE model and the modified version of C-KASE, on our WSD datasets. The green represents the results of the original C-KASE method, and the blue represents the modified version, which we refer to as C-KASE'. 38

2.6 The accuracy comparison of C-KASE and its modified version on the four-word types of nouns, verbs, adjectives, and adverbs. The green represents the results of the original C-KASE method, and the blue represents the modified version, which we refer to as C-KASE'. 39

2.7 Accuracy comparison of 1-NN WSD evaluation framework on the unified dataset, using BERT, LMMS, SensEmBERT, ARES as baselines and our method, C-KASE. 40

2.8 Comparing the accuracy of the 1-NN WSD of the four considered embeddings as baselines, and our method, C-KASE. The dataset in this experiment is a concatenation of all five datasets, which is split by Part-of-Speech tags. 43

3.1 Demonstration of the Wikification problem. Text T is our input text with a list M of ambiguous mentions. The task is finding E, including the correct chosen entity e_i for each mention. 56

3.2 Partitioning the extracted mentions of sentence "Mars, galaxy, and bounty are all chocolate" into separate windows of size 3 in (a), and partitioning it into overlapping windows of size 3 with offset 1 in (b). 63

3.3 Segmenting the text into overlapping windows. In this figure, our text has 24 mentions. The considered window's size is 20, and the Offset size is 1; it means two consecutive windows overlap 19 common mentions, and we have 5 windows in total. 64

3.4 The effect of Window Size |WS| on CACW's accuracy, using "Wiki30000" dataset, with different offset (OF). In a) the number of senses for each mention is set to 5. In b), the number of senses for each mention is 10, and in c), the number of senses for each mention is set to 15. . 71

3.5 Run-time comparison of the Wikisim approach and CACW approach, by changing window size; WS= 3, 5, 10, 15, 20, 25, 30, 35, 40, and offset=1, each run. In (a), we compare the models on the short texts, while (b) represents the results of comparing the models on the averaged length text, which is Wiki5000. The subfigure (b) on the right displays the results of comparing the models on the larger dataset, the Wiki30000. This result shows that while the new algorithm's accuracy increases, the run-time slightly increases compared to the Wikisim algorithm. 76

4.1 Word clouds for the tweets that present each kind of harassment content. 86

4.2 Classifiers learning curve on a k-fold cross-validation over the validation set with 45 TF-IDF features. 91

5.1 Dataset Statistics. 'Indirect H.' stands for indirect harassment, 'Physical H.' refers to physical harassment, and 'Sexual H.' means sexual harassment. The first figure shows the statistics of each harassment class in the training data. The second figure represents the statistics of the validation dataset and number of instances in each class. The last figure represents the statistics and number of instances in each class of the test dataset. 111

5.2 Accuracy heatmap of each one of the models (in rows) on different harassment types (in columns). The lighter color indicates lower accuracy. 115

C.1 Python code snippets on how to calculate the weighted overlap score between two NASARI vectors. 149

C.2 Python code snippets on weighted overlap measure for concept "cat". 150

Abstract

This dissertation presents a new method for text representation learning and applies it to two Natural Language Processing (NLP) problems, namely, word sense disambiguation and text classification. Word Sense Disambiguation (WSD) is a problem in NLP when there are different possible meanings for words present in the text. These possible meanings are extracted from a knowledge base. The correct meaning of a word in the text can be identified based on surrounding words and prior knowledge. When Wikipedia serves as the knowledge base, this problem is referred to as Wikification. We provide two algorithms for solving the Wikification problem by segmenting the text and assigning weights to different meanings of a word based on their context's relevancy. For the WSD problem, we study the role of representation learning in the final output of the WSD algorithm and incorporate our novel representation learning approach. We use our method when solving the WSD problem with the 1-nearest-neighbor algorithm and demonstrate that our representations work better than the state-of-the-art models in the WSD task. We evaluate our novel representation method on general English and biomedical texts. The results demonstrate that, by considering context from various sources in representations, the results of the WSD task can be improved.

Text classification is the second NLP problem that we study. We consider a collection of tweet posts and classify them into two groups of tweets, harassment versus non-harassment. This binary classification task is addressed with standard supervised methods. Next, we focus on categorizing harassment tweets into specified harassment types, for which we combine our novel text representation with a graph convolutional network. In experiments, we demonstrate the effectiveness of our approach by comparing it with other language models and classical representation models.

Acknowledgements

First and foremost, I would like to thank my supervisors, Prof. Evangelos Milios and Prof. Norbert Zeh, for their valuable support, advice, and comments. They are not just my Ph.D. advisors but also outstanding role models who taught me never to hesitate to take on a new challenge. Their dedication to research and academic service is admirable to me. I am truly honored that I finished my Ph.D. under their supervision. They generously provided me with many perfect chances for collaborations through which my Ph.D. experience became remarkably more educating and exciting.

I also learned a lot from a group of extraordinary researchers I was fortunate to collaborate with during my Ph.D., including Prof. Kaveh Mahdaviani, Prof. Vlado Keselj, Prof. Lilian Berton, and Prof. Travis Gagie. In particular, I owe much to Kaveh for his generous and caring support over the last four years. He is a great mentor and an exceptional researcher who contributed a lot in guiding me through my Ph.D. from the moment we started to collaborate. Kaveh was like a third supervisor for my thesis who was always present whenever I needed advice or had an idea to discuss. He has been a wise and trusted advisor, helping me to think through my ideas and arguments logically. I like to thank the Mila Institute and the Machine Learning and AI engineering group at the Vector Institute in Toronto, particularly Mr. Ron Bodkin for providing me with an exceptional internship experience during my PhD. I learned a lot of useful skills during this internship.

I extend my heartfelt gratitude to my esteemed Ph.D. committee members, Prof. Evangelos Milios, Prof. Norbert Zeh, Prof. Ana Gabriela Maguitman, Prof. Ali Ghorbani, Prof. Vlado Keselj, Prof. Jeannette Janssen, Prof. Travis Gagie, Prof Marilyn Macdonald, and Prof. Enayat Rajabi, for their invaluable guidance and insightful feedback. Their contributions have significantly strengthened the present version of my thesis. Their insightful comments and probing questions not only illuminated previously unseen facets of my research field but also unlocked new avenues for exploration and deepened my understanding. This enabled me to refine my work and present it in a more comprehensive and impactful manner.

I owe a deep gratitude to Professor Gagie, whose singular advice, encompassing both

academic and personal spheres, has been a constant source of encouragement. Having the privilege of serving as a Teaching Assistant for his course on algorithms design and analysis provided me with invaluable insights into the immense effort involved in crafting a successful learning experience. Additionally, I extend my heartfelt appreciation to all other faculty members who entrusted me with the opportunity to assist them as a Teaching Assistant across various courses in the Computer Science and Mathematics departments. These include Professors Brodsky, Whidden, Ralph, Siegel, Hawkey, McAllister, He, Keselj, Sampalli, and Dr. Dey. I thank all the people in the Computer Science department, including Vidhya Ramamoorthy, the IT center technician, as well as Helena Martel from the faculty of graduate studies.

I wish to thank my family for their unconditional love and support and for giving meaning to everything in my life. I am also grateful to my friends Sarah Boyd, Dijana Kosmajac, Stacey Taylor, Samuel Bruno da S. Sousa, Noreen Smith and her beautiful family, and Serikzhan Kazi for all his guidance and for being a supportive friend during the nights and days working in the lab.

Finally and above all, I cannot begin to express my unfailing gratitude and love to my beloved husband, Kaveh, who has supported me throughout this process and has constantly encouraged me when the tasks seemed arduous and insurmountable. You are the light of my life, helping me find my way through the darkness; no words could genuinely convey how appreciative I am for having you in my life. I would not have been able to come this far if it were not for your endless love and support. You are the shoulder that I lean on to find peace and harmony; thank you, and love you, my wonderful Kaveh.

This thesis is based on the following publications:

A) Papers of Representation Learning and WSD Problems:

- Biomedical Word Sense Disambiguation with Contextualized Representation Learning, Mozhgan Saeidi, Evangelos Milios, Norbert Zeh, The International Workshop on Semantics-enabled Biomedical Literature Analytics (SeBiLAn), In Companion Proceedings of the Web Conference 2022 (WWW'22 Companion), Pages 843–848, April 25–29, 2022, Virtual Event, Lyon, France. ACM, https://doi.org/10.1145/3487553.3524703.

  The above is an extended and elaborated version of the work that previously appeared in:

- Contextualized Knowledge Base Sense Embeddings in Word Sense Disambiguation, Mozhgan Saeidi, Evangelos E. Milios, Norbert Zeh, Workshop on Machine Learning (WML 2021, 3rd edition) 2021, 16th International Conference on Document Analysis and Recognition (ICDAR), Pages 174-186, 5-10 September 2021, Lausanne, Switzerland, https://dl.acm.org/doi/abs/10.1007/978-3-030-86159-9-12.

- ContextBERT: Contextual Graph Representation Learning in Text Disambiguation, Mozhgan Saeidi, Machine Learning with Symbolic Methods and Knowledge Graphs, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2021), Pages 15-29, September 13-17, 2021, Spain, https://ceur-ws.org/Vol-2997/paper2.pdf.

B) Paper of Wikification Problem:

- Context-Enhanced Concept Disambiguation in Wikification, Mozhgan Saeidi, Kaveh Mahdaviani, Evangelos Milios, Norbert Zeh, Intelligent Systems with Applications Journal, 200-246, 2023.

C) Papers of Harassment Categorization Problem:

- Graph Convolutional Networks for Categorizing Online Harassment on Twitter, Mozhgan Saeidi, Norbert Zeh, Evangelos Milios, 20th IEEE International Conference on Machine Learning and Applications (ICMLA 2021), Pages 946-951, December 13-15, 2021, California, https://ieeexplore.ieee.org/document/9680133.

- Categorizing online harassment on Twitter, Mozhgan Saeidi, Samuel Bruno da Sousa, Evangelos Milios, Norbert Zeh, and Lilian Berton, SocIaL Media And Harassment (SIMAH), Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2019), Communications in Computer and Information Science, vol 1168, Pages 283-297, September 2019, Germany, https://link.springer.com/chapter/10.1007/978-3-030-43887-6-22.

Chapter 1

Introduction

Text has been used in our communications and storing humankind's knowledge and literature for thousands of years. Our daily communication and activities for different purposes frequently occur through various computing machines and online platforms via text. The usage of computers and these platforms is rapidly growing. For example, we use text on social networks, send emails, and write reports, to name a few. Multiple meanings might be inferred from the text, and not all are necessarily correct in conveying the true meaning. Humans understand the text's correct meaning based on some extra information, including shared knowledge stored in knowledge bases and encyclopedias such as Wikipedia. This extra information, which we refer to hereafter as *context*, can also partly come from the rest of the text. Considering our usage of computers in our daily lives, it would help if computers could apply near-human understanding analysis on text written in human language. For example, the number of users worldwide would dramatically increase when computers can process different languages well. So, if computers can understand the text correctly and completely, then it is possible to expect correct results from the analysis of the text by computers. Therefore, it is crucial to enhance the machines' ability to understand the text correctly, like humans. This thesis focuses on the semantic understanding of the text.

In this thesis, first, we propose a novel model in semantic representation learning that considers the context in its representations. This context comes from multiple sources, including the knowledge bases and other parts of the input text. To evaluate the power of this new method, we apply the proposed semantic text representation learning method to two fundamental NLP tasks, namely "word sense disambiguation" and "text classification". In the following, Section 1.1 provides an overview of this thesis's contribution to the text disambiguation task, and Section 1.2 provides the overview of this thesis's contribution to the text classification task.

In Natural Language Processing (NLP), many different forms and approaches for representation learning (embedding) have been suggested. For example, BERT [38] and

SBERT [163] are two recent state-of-the-art language models that consider surrounding words of each ambiguous word as extra information when generating the vector representation. However, suppose the numerical representations of the text do not carry sufficient information about that text. In that case, the machine's algorithm is not able to understand the text correctly and so cannot provide good results in different NLP tasks, such as machine translation [225], summarization [102], and part of speech tagging [111], to name a few. So, representation learning has an essential effect on the final result of each NLP task [199]. This thesis introduces a new approach to the representation learning problem. This approach has the novelty of combining contextual information from multiple sources in its vector representations. These sources include knowledge bases and contextual information from the input text. Combining this information as context when generating our vector representations differentiates this method from other state-of-the-art methods. This combination happens via concatenating the vector representation of the extracted information from the knowledge base and the vector representation of the surrounding words of each word that is considered ambiguous. We consider recent state-of-the-art contextual language models as baselines. The results of comparisons with these baselines show how well our method works in various NLP tasks, including word sense disambiguation and text classification.

1.1 Text Ambiguity

Words are one of the building blocks of a text that we consider in this thesis. Some words have multiple meanings. For instance, the word "bank" could refer to both a financial institution and the land alongside a river. The words with multiple meanings are called *ambiguous words*. When we have an ambiguous word in a text, humans can think of the various meanings of that word based on their knowledge. Then, they try to find the ambiguous word's correct meaning based on the text's neighboring words. All these surrounding words, alongside our knowledge and other information about the text, provide intelligence that leads us toward the correct meaning. We use the name "context" to refer to these neighboring words of the input text and other words of the text and our knowledge. When we have ambiguous words in a text, we call it an ambiguous text. In NLP, word sense disambiguation (WSD) is a well-known task that tries to find the text's true meaning, leading to a more accurate understanding of the text. We work on this NLP task toward our goal of semantic understanding of the text in chapters 2 and 3 of this thesis.

It is shown in the literature that considering the context in an embedding approach improves the quality of the representations for many NLP tasks, including the word sense disambiguation task [199]. As mentioned above, we can consider the surrounding words of each target word in a text as its context. In addition, we also have knowledge bases and thesaurus, such as Wikipedia and BabelNet [139], which comprise information on words and concepts related to each word. The information stored in the knowledge bases can be considered alongside each word in a text when we convert words into numerical vector representations for computers. Considering and combining these two sources of information (one is the surrounding words, and the other is information from knowledge bases) as context for the new proposed embeddings is the novelty of my work. I use context in the proposed semantic representation learning model when generating the representations. Concatenating the context from these two sources makes this new approach different and unique from previous works in the literature. Running multiple experiments, we show our representation vectors enable the 1-Nearest Neighbor algorithm to disambiguate text significantly better than recent context-based language models as our baselines in terms of accuracy.

We expand our experiments showing the applicability of our approach to both general English text and Biomedical text as an application.

1.2 Text Classification and Categorization

In the second part of this thesis, I study the problem of categorizing tweets that include harassment and then find the type of harassment as a text classification problem. Based on the Oxford dictionary, "Harassment is the act of annoying or worrying somebody by putting pressure on them or saying or doing unpleasant things to them"[1], either verbal or physical. For this online harassment classification task, I consider binary classification, i.e., whether a tweet includes harassment or not, and multi-class classification, i.e., which type of harassment is present in a tweet. The types present in our dataset are divided into three classes: indirect harassment, physical harassment, and sexual harassment. To show the importance of vector representations in solving this harassment classification problem, first, I study using TF-IDF and Word2Vec features in multiple supervised machine learning (ML) algorithms as baselines. The results help us match the features with the

[1] https://www.oxfordlearnersdictionaries.com/definition/english/harassment

supervised ML method for better classification accuracy. Second, I was interested in studying the impact of deep learning approaches to compare the power of these recent methods versus classical machine learning approaches on this classification task. I create and use a Graph Convolutional Network (GCN) model, which is a new approach to solving this problem in the literature using our new embeddings. A graph convolutional network model is an approach for semi-supervised learning on graph-structured data. To set this up, we consider each tweet post as a node and connect a node with its nearest neighbor. The distance measures are cosine similarities of the node's vector representations. We try different embedding methods and compare the results of classification. This deep learning method and pre-trained embedding models significantly improve online harassment classification compared to the baselines. We conjecture that the improved classification performance is due to using the neighbors' information in the contextual embeddings via the graph structure of the data points.

1.3 Representation Learning Development

In multiple works around 1990, it was observed that numerical representations could be generated by utilizing context features. During that time, in information retrieval, latent semantic analysis was a model that showed dramatic changes in its results using this representation. It was one of the early approaches to using context in embeddings, which have continued till today in deep learning models. In this thesis, for the aim of semantic representation of the text, first, we analyze state-of-the-art approaches with respect to the word disambiguation task. These methods include BERT [38], LMMS [104], SensEmBERT [180], and ARES [181], that are recent transformer-based language models in this task. We use these representations and disambiguate documents using the 1-Nearest Neighbour algorithm. We observe the pros and cons of the methods by analyzing the errors of each embedding method at the time of disambiguation on part-of-speech tags. This analysis shows that current contextual representation learning methods, such as BERT, have shortcomings in disambiguating particular parts of speech, like verbs. We show how these models suffer from the lack of contextual information, like the meaning of each word from the knowledge bases. Based on these analyses, we propose a new semantic representation learning method attempting to mitigate the negative aspects of current methods. This model considers the information from knowledge bases and the information from input text in each word's

representations. To evaluate our new proposed semantic representation learning method, we apply it to two NLP tasks. The first task is text disambiguation, and the second is classification.

1.3.1 Text Disambiguation Solutions

We consider two specific problems for the text disambiguation task: word sense disambiguation and Wikification. In word sense disambiguation, we need to find the correct meaning of ambiguous words between a set of possible meanings. The results of this analysis of the text disambiguation problem are presented in the first part of Chapter 2. Based on this result, which shows the importance of context in embeddings, we bring the idea of adding context from knowledge bases into embeddings in a new approach. We evaluate this method versus other baselines on general English text. To investigate further applications of our method, we explore Biomedical text. For the Biomedical domain, as baselines, we consider BioBERT [93], Bio Graph [40], and DeepBioWSD [152], as they are recent contextual language models that have been used for WSD on biomedical texts. The presented results of our approach's comparison with these baselines show how well our method works in this specific domain and general English text.

Wikification is one specific example of a text disambiguation task when the knowledge base is Wikipedia. This Wikification task has two parts: first, detecting ambiguous words, and second, finding those words' correct meaning. This thesis presents two new algorithms for this second part of the Wikification task as a special example of the WSD problem. Our algorithms include the idea of text segmentation and weighing of possible meanings of ambiguous words. When we compare the results of these algorithms with the recent baselines, we observe between 10-20% improvement in the accuracy measure in finding the correct meanings on multiple datasets.

1.3.2 Text Classification Solutions

For the text classification task, we use our new representation learning model in a graph convolutional network as a deep learning approach toward solving the online harassment classification problem. To build a solid baseline to compare our model, we use transformer-based language models, BERT and SBERT, and two classical embedding approaches, TF-IDF and Word2Vec. We complement these techniques with supervised ML approaches.

In this online harassment classification task, we demonstrate that the supervised machine learning model exhibits greater adaptability when using the TF-IDF and Word2Vec feature sets.

To show the vital role of representation learning in harassment classification with current methods, we use the graph convolutional networks method [224]. We build a graph with the set of tweets and use our new representation learning method as the embeddings. Comparing the classification results using GCN with our embeddings and BERT embeddings shows that our method improves accuracy above 91% in this online harassment categorization task. We also evaluate the role of this GCN deep learning method in solving this classification task and compare it with FastText [13], LSTM [62], and CNN [92] methods. The results present how this deep learning model works better than other recent methods.

1.4 Goals and Contributions

The goals of this thesis include the following:

- Analyzing and investigating different representation learning approaches, understanding how they work, and evaluating their effects on two related NLP tasks; WSD and text classification. Designing techniques that incorporate context (in different forms) into embeddings. Proposing a novel vector representation learning method that reduces the shortcomings of current embedding approaches while preserving their advantages.
  - Studying the problem of word sense disambiguation and current approaches for this task. Evaluating our new representation learning method on the WSD task and comparing the results of disambiguated words on parts of speech.
  - Investigating the text classification problem by recent deep learning approaches and comparing the results with classical ML methods. Considering various embedding methods as features of the classifiers and showing representation's role in the final results of the classifier.
- Proposing novel algorithms for the task of Wikification; the task of Wikification considers Wikipedia as the knowledge base when disambiguating ambiguous words. We investigate some of the previous approaches and compare the results of our method

with these baselines. Our comparisons prove that the novel idea of segmenting the text and assigning weights would improve accuracy compared to the baseline methods.

Most of the code developed in this thesis is open source and can be publicly accessed on the author's Github page [2]. The exact Github repository of each project is addressed at the beginning of each chapter.

1.5 Thesis Organization

We can divide this thesis into two parts; the first part includes our work on the text disambiguation task in chapters 2 and 3, and the second part includes our contribution on the text classification task in chapters 4 and 5. The detailed organization of the thesis is as follows:

- Chapter 2 focuses on text representation learning. It starts with analyzing the performance of current state-of-the-art embedding approaches and their pros and cons, giving a better view to design a new representation learning model. Based on the results of this analysis, we provide a novel sense embedding method designed to improve the defects of the prior embedding models. The results of comparing this method with other state-of-the-art methods on general English text and Biomedical text are presented in this chapter, which shows the potential of this approach.

- Chapter 3 is our study on the Wikification problem. We propose two algorithms for solving link ambiguity based on the keyword selection baseline approach [173]. In the comparisons with different approaches, we show how important the role of context is in dealing with text ambiguity, as demonstrated by our algorithm's results. Furthermore, we demonstrate how we can rank a candidate's senses based on their relevance to the context and then choose the correct meaning for the candidate.

- Chapter 4 provides a detailed description of the online harassment classification problem we are trying to solve. The dataset that we consider in this task is introduced in this chapter. This chapter aims to build a strong baseline to compare the results of classical machine learning models on this classification task. So, we outline the process of categorizing text, including instances of harassment on social media, through the application of supervised machine learning methods that take into account

[2] https://github.com/mozhgans

various text features. We specifically show the applicability of these approaches to classify a dataset collected from Twitter. The results show which supervised ML model works best with each text's feature for online harassment classification task.

- Chapter 5 contains our contribution to solving this online harassment classification problem using different embeddings and deep learning approaches. We build a graph convolutional network model to solve this tweet classification problem and show how this model performs well with our embeddings as well as other context-based embedding approaches. We further provide detailed results comparing classical machine learning and deep learning approaches to address this problem.

- Chapter 6 is the concluding chapter, including a summary of the results and a discussion of possible directions for future research.

Chapter 2

Contextual Knowledge-Base Representation Learning in Text Disambiguation

Contextual word embedding has been shown to carry useful semantic information to improve the final results of various Natural Language Processing (NLP) tasks [228]. However, it is still challenging to integrate these embeddings with the information of the knowledge bases. This integration is helpful in NLP tasks, specifically in the lexical ambiguity problem [155]. Word Sense Disambiguation (WSD) is one of the main problems at the core of the NLP domain [11]. The WSD problem arises when we encounter ambiguous words in a text and need to determine their correct meanings. Text representation is a crucial aspect of all WSD models [34]. It encodes the text and relevant information to determine the most appropriate meaning for disambiguating the text [14]. While various approaches exist for context-based representation learning, there is a lack of studies examining the pros and cons of encoding word senses [106]. In this study, we show the effectiveness of transformer-based language models in the task of WSD by providing an in-depth quantitative error analysis of these contextual embeddings in the WSD task. We use different recent contextual embedding methods and disambiguate the document by applying a simple nearest neighbor (k-NN) approach. We analyze the errors that occur with each model through part-of-speech tags. In analysis with part-of-speech tags, we first introduce quantitative error rate formulas to formulate the errors, and second, we use a confusion matrix to complete our analysis. Our experiments show that using context-based embedding results in a more coherent text document after disambiguation. Furthermore, we show that BERT-based pre-trained language models' accuracies vary in disambiguating different parts of speech. After analyzing the current language models and investigating their pros and cons, we propose a new embedding approach that considers the information from the context (the input text) and the information from the knowledge base. The name of this approach is C-KASE (Context-Knowledge base Aware Sense Embedding), a novel approach to producing sense embeddings for the lexical meanings within a lexical knowledge base. C-KASE

representations enable a simple 1-Nearest Neighbor algorithm to outperform state-of-the-art models in the English WSD task. Since this embedding is tailored to each knowledge base, it outperforms similar tasks that rely on specific knowledge bases, i.e., Wikification and Named Entity Recognition. Our experiments provide proper settings for the C-KASE representation learning model comparable to supervised and knowledge-based approaches. The results of comparing our approach with current state-of-the-art models show the effectiveness of our method [1].

2.1 Introduction

Natural Language Processing (NLP) encompasses various tasks, and multiple studies have demonstrated that representation learning significantly influences NLP task outcomes [51]. Word Sense Disambiguation (WSD) is among these tasks influenced by the chosen embedding approach. WSD refers to the problem of determining the correct meanings of words with multiple possible meanings within a text. In the literature, a word with one meaning is called monosemous, while a word with multiple possible meanings is called an ambiguous word [207]. Each possible meaning of an ambiguous word is called a word sense. Finding and matching these ambiguous words to their correct meaning (sense) in the text is called the word sense disambiguation task in NLP. These different meanings for ambiguous words have been collected in a predefined list of senses (meanings). We refer to these lists in the NLP domain as the knowledge base or sense inventory. Recent pre-trained embedding methods, such as ELMO [153], BERT [38], and XLNET [223], have demonstrated vital roles in the WSD task [209].

Different approaches have been applied to solve the WSD problem. We can categorize these approaches into two broad categories, supervised and knowledge-based approaches [137]. In the first category of supervised methods, algorithms rely on semantically annotated corpora for training [86]. In the second category of knowledge-based approaches, the algorithm depends on the structure of a semantic network. In this category, the algorithms try to investigate the graph structure of the knowledge base to find a list of possible meanings for each ambiguous word in the text.

While the recent contextual language models have shown success on the task of WSD, we still need more studies analyzing these language models to understand their behaviors

[1] Part of this chapter is published in [168], [166], and [171].

more accurately [24]. Especially in the task of WSD, we need to understand these language models carefully and analyze their behavior on each word type. In this regard, recently, Loureiro et al. tried to track the behavior of language models based on different layers of the model [106]. In our work in this chapter, we follow two goals. First, we aim to analyze the behavior of recent context-based language models and investigate their pros and cons. Second, to address the limitations of current embedding approaches, we propose a new embedding approach that improves upon current embedding approaches by addressing their shortcomings. We show our approach is effective in solving the task of WSD in multiple experiments. Finally, following our first goal, we identify which parts of speech are most commonly disambiguated incorrectly by different methods. After these analyses, we focus on the error analysis of word categories and suggest some possible future directions. All the code and data are publicly available [2].

2.2 Related Work

In the following, we provide a general background of semantic similarity methods, language models, and WSD approaches. Then, we continue this chapter by delivering our new proposed embedding model and showing how it solves the WSD problem.

2.2.1 Semantic Similarity

In NLP, different metrics are concerned with document similarities. These metrics measure how similar the meanings of the two documents are. Words are represented as vectors in a high dimensional vector space such that words with similar vectors have similar meanings. Multiple mathematical tools are defined in the semantic similarity domain to calculate the distance between the words. For example, the vector space model is one tool for this distance/similarity measurement [52]. Some examples of these commonly used similarity measures are Cosine Similarity and Euclidean Distance. These methods are widely used in various NLP applications such as text classification [117], information retrieval [61], machine translation [219], and many more.

We need to convert text as an unstructured data type into a numerical representation; this numerical representation is helpful when designing algorithms to allow a machine to

[2] https://github.com/mozhgans/Error-analysis-of-concept-embedding-Approaches

work with text as input data (for example, it is helpful in increasing algorithm robustness). To develop effective novel representation learning algorithms and address various NLP tasks, such as word sense disambiguation and text classification, it's essential to first comprehend word embedding approaches and their impact. In semantic similarity methods, text data must be transformed into vector representations of text features. The quality of the resulting similarity scores heavily relies on this vector representation. Many semantic similarity methods, like cosine similarity and Euclidean distance, integrate well with word embedding techniques, leading to promising results [36]. Therefore, gaining a comprehensive understanding of word embedding methods and their effects is a crucial step in these endeavors.

2.2.2 Text Representation Learning

In most machine learning and NLP tasks, when we work with data, we want to recognize and predict any patterns in the dataset, if possible [194]. Representation learning is important because it converts high dimensional data into low dimensional data [9]. When working with lower dimensions, it is easier to recognize the patterns and predict the behavior of the data points. In NLP, this means learning a lower-dimensional representation of raw text data.

Word Embedding

In word embedding, when we convert the words into a numerical vector representation, the representation of words with closer meaning is closer in the vector space. One basic word embedding method is "Term Frequency Inverse Document Frequency" (TF-IDF) [178, 176, 183, 27]. TF-IDF is a numerical statistic that is used to reflect the importance of a word in a document or a set of documents. It is often used as a weighting factor in information retrieval and text mining. The TF-IDF value increases proportionally to the number of times a word appears in the document. However, it is offset by the frequency of the word in the entire corpus of documents so that common words such as "the" and "is" are not given too much weight [27]. To have an embedding with high quality, we need to have this learning happen on a large amount of text; it can be learned alone or jointly during word embedding model training. The former is good when we use the same learned embedding for the other tasks. The latter is suitable when the embedding is part of another task. When working with these numerical representations of words, we want to have the semantic information in the

word vectors. Most representation learning methods recently aim to consider this need [54]. Word2vec [120, 121] and GloVe [151] are two of these approaches. These methods use a co-occurrence frequency matrix of words for the vector representations of each word. For these two models and almost all word embedding approaches, the goal is to measure the semantic similarity of various parts of the text, in different tasks, like machine translation and word sense disambiguation [99, 27]. One of the most critical challenges in each text embedding approach is meaning conflict [24], which refers to a situation in which there needs to be more agreement or understanding about the meaning of a word, phrase, or concept. For example, consider the word "bank" that can have multiple meanings, such as a financial institution in the sentence "I need to visit the bank to withdraw some money", or it could refer to a river bank as in the sentence "I went for a walk along the bank of the river". It is crucial to enhance the representation so that the correct meaning for each word or phrase is considered in the generated embedding [95, 129].

Two broad categories of representation learning techniques are contextual representation learning and non-contextual representation learning methods. Non-contextual representation learning techniques aim to learn representations of data independent of their context. For example, each word is assigned a fixed-size vector representation in a word embedding model such as Word2Vec [120, 121] based on its co-occurrence with other words in a large corpus of text. The resulting vector representation of each word is the same, regardless of the context in which the word appears. In contrast, contextual representation learning techniques aim to learn representations of data dependent on the context. For example, in a contextual word embedding model such as BERT [38], each word is assigned a vector representation that varies depending on the surrounding words in the sentence. The context-dependent representations capture the meaning and relationships between words in different contexts. For example, the vector representation of the word "bank" will be different for its different word senses. In the context of a financial institution, the vector representation will be closer to the vector representations of other words related to finance, such as "money" and "loan". In the context of a riverbank, the vector representation will be closer to the vector representations of other words related to geography, such as "river" and "shore".

Some hybrid approaches generate semantically aware word representations. NASARI

vector is one of these approaches [26], which stands for New Approach to Semantically-Aware Representation of Items. In this approach, BabelNet [3] is the knowledge base, and this approach considers two types of concept relations of hypernyms and hyponyms [139]. *Hypernyms* are words with a broader meaning than a specific word. For example, "animal" is a hypernym of "dog". *Hyponyms* are words with a more specific meaning than a general word. For example, "Golden Retriever" is a hyponym of "dog". NASARI vectors are created using the following steps. In the first step, it gathers Word2Vec embedding vectors for the target concept and all its related words through hypernyms and hyponyms relations, which are extracted from BabelNet [4]. In the second step, it computes the cosine similarity between each word embedding and the target concept [88, 25, 27]. The final representation for the target concept is a vector which includes the calculated cosine similarities. The similarity measure between two NASARI vectors is the Weighted Overlap (WO) measure [156]. The weighted overlap is a measure of similarity between two vectors, that takes into account the relative importance of each dimension. This measure takes into account the relative importance of each dimension by weighting the cosine similarity scores by the absolute rankings of the dimensions in the two vectors. This ensures the dimensions that are more important to the meaning of the vectors have a greater impact on the similarity score. This measure also takes into account the lexical specificity of the dimensions by using the inverse of the rank of each dimension. This means that dimensions that are associated with more specific words or concepts are given a higher weight. WO is defined as follows:

$$WO(v_1, v_2) = \sqrt{\frac{\sum_{d \in O}(rank(d, v_1) + rank(d, v_2))^{-1}}{\sum_{i=1}^{|O|}(2i)^{-1}}} \tag{2.1}$$

In the above equation, v_1 and v_2 are the NASARI vectors for the two concepts to be compared. O is the set of overlapping words in the vectors v_1 and v_2, d is a word in the set O, and $rank(d, v_i)$ is the rank of the word d in the vector v_i, based on the specificity described above [25]. The rank of a dimension in a NASARI vector is simply its position in the vector. The rank of a dimension can be used to interpret the meaning of a NASARI vector. For example, if two NASARI vectors have a high overlap in dimensions with high ranks, then this means that the two concepts are very similar. The rank of a dimension can also be used

[3] BabelNet is a multilingual encyclopedic dictionary with a wide coverage of terms; https://babelnet.org/about

[4] Brief information on BabelNet with an example is provided in Appendix B.

to improve the performance of NLP tasks that use NASARI vectors. For example, in a word similarity task, the similarity score between two words can be weighted by the ranks of the overlapping dimensions. This ensures that dimensions that are more important to the meaning of the words have a greater impact on the similarity score. For more details and an example implementation of the WO measure in Python please refer to Appendix C.

2.3 Word Sense Disambiguation

Our focus in this chapter is on the task of Word Sense Disambiguation (WSD). The complexity of this problem depends on multiple sources. One of these sources is the need for representation learning. As discussed above, there are various word and text representation learning methods, and each has unique dependencies and difficulties. The other source of complexity is the dependency of the problem on the knowledge base. Therefore, it is crucial to consider the proper knowledge source, depending on the use case and the text domain subject. Considering the adequate knowledge base provides the opportunity to collect appropriate senses from the knowledge base [49]. For example, consider disambiguating a biomedical text; if the knowledge base covers general concepts rather than biomedical terms, the accuracy of correct disambiguation would be lower than when the knowledge base covers all the biomedical concepts.

In a text document, there might be words or phrases with multiple meanings, and we as humans will understand the correct meaning based on the context of the text, which is the surrounding words. In WSD, the algorithm enables the machine to identify this true meaning. The multiple meanings of ambiguous words are called senses. Therefore, there should be a source of knowledge covering the words with their different meanings, which we call a knowledge base or sense inventory. If we consider these various senses as classes and try to assign the correct meaning (sense) to each word, then WSD is an example of a classification task [137].

We can generally consider four main sub-tasks to design a WSD system. These sub-tasks are as follows:

- Candidate selection (i.e., word senses)
- Knowledge base selection
- Context representation

- Selection of the classifier

Various word sense inventories include different senses of words. In each research work on the WSD task, one of the critical actions is choosing this word sense inventory [20]. So, paying attention to which external knowledge source we use in the task is important.

2.3.1 External Knowledge Sources

Knowledge sources (lexical resources) are fundamental sources of information in the task of WSD. They provide a set of meanings (senses) for the ambiguous words we try to disambiguate. There are different types of knowledge sources, depending on the subject of the text and the relations between the words. Some examples include thesauri, machine-readable dictionaries, ontologies, and corpora. Among the widely used knowledge sources are Wikipedia, WordNet [122, 44] and SemCor [124].

2.3.2 Context Representation

As discussed above in Section 2.2.2, we need to apply some pre-processing steps to the text data to make it suitable for the machine [147]. Some pre-processing steps include tokenization, part of speech tagging, and lemmatization [141]. These pre-processing steps aim to convert the text into numerical featured vector representations.

2.3.3 Selection of the Classifier

We can consider two main machine learning-based algorithm types for solving WSD; Supervised learning WSD and Unsupervised learning WSD methods. Besides the machine learning-based approaches, knowledge-based approaches are the other methods to solve the WSD problem.

Knowledge-based Approaches

The structure of the knowledge base plays an essential role in this type of approach; therefore, WSD is a graph-based problem [137]. SensEmBERT [180] is one of the latest methods in this category that combines knowledge from a semantic network with a language model. Following this approach, ARES [181] provides sense embeddings in a lexical knowledge

base. Unfortunately, despite using powerful knowledge bases, the accuracy of knowledge-based approaches is almost always less than supervised approaches [10].

Supervised Approaches

The supervised approaches usually use sense-annotated data at the time of training. These approaches mostly use neural networks architectures [115], or support vector machine (SVM) models [67]. These methods normally achieve the best accuracy. Some approaches consider the WSD problem as a translation task, where the input is a word in its context, and the output is a sequence of senses for that word [158]. This approach utilizes neural machine translation models, such as encoder-decoder architectures, to learn the mapping between the input word and its possible senses [201]. The major drawback is that it requires large amounts of labeled data for training the model. On the other hand, some other works showed the potential of contextual representation learning in WSD task [115, 153], such as NASARI vectors [24], LLMS vectors [104], and ARES vectors [181]. In NASARI (Novel Approach to a Semantically-Aware Representation of Items) vectors, the representations are based on structural knowledge extracted from a multilingual semantic network [24]. LMMS (Language Modelling Makes Sense) vectors consider sense-level embeddings with complete WordNet coverage and show the power of this representation for WSD by applying a simple Nearest Neighbors (k-NN) method [104]. ARES (context-AwaRe Embeddings of Senses) used the 1-NN method with its representations and showed improved results in the disambiguation task [181].

The k-nearest neighbors (k-NN) method in WSD uses labeled training data to learn the relationship between the features of the text and then applies that knowledge to classify new, unseen instances of the words based on their features. The k-NN algorithm finds the k training examples closest (most similar) to the input and then assigns the input the most common label among those k examples. K is always a positive integer that can be determined experimentally. For example, in the WSD task, the 1-nearest neighbor matching is one of the simple ways to disambiguate a word [58]. Using this k-NN method to find the nearest vectors similar to word senses to solve the WSD problem has been used in many works based on contextual word representation learning approaches [115, 153]. We can use various distance metrics to calculate the distances between the test vector and training vectors. Euclidean distance and cosine distance metrics are the most used metrics in most

NLP tasks [56].

Unsupervised Approaches

Unsupervised word sense disambiguation is another technique for determining a word's correct sense in a given context without using labeled training data. Instead, these methods typically rely on statistical or machine learning techniques and use external sources such as WordNet or Wikipedia [161, 150]. Some examples of unsupervised WSD methods include:

- Distributional methods: These methods use distributional semantics to infer the sense of a word based on its co-occurrence patterns with other words in a corpus [150].
- Clustering methods: These methods group words into clusters based on their similarity in context and then use these clusters to disambiguate word senses [110].
- Graph-based methods: These methods represent words and their contexts as nodes in a graph and use graph-based algorithms to disambiguate word senses [3].
- Neural methods: These methods use neural network architectures like RNNs and transformers to disambiguate word senses [165].

It is important to note that unsupervised methods generally tend to have lower accuracy than supervised methods. Still, they have the advantage of not requiring labeled data.

In the following section, first, we evaluate the performance of some of the recent context-based embedding approaches on the WSD task. Then, we analyze the errors of each of these models and evaluate their performances. Our goal for this evaluation is to understand the pros and cons of these recent embedding models and to build a new representation learning model that reduces the defects of the current approaches.

2.4 Error Analysis Framework

We consider four different state-of-the-art contextual embeddings in the WSD task using the nearest neighbor approach and analyzing each method's errors. The embedding approaches that we consider here are BERT [38], LMMS [104], SensEmBERT [180], and ARES [181], as the most recent contextual embedding models in the task of WSD. Choosing this simple 1-Nearest Neighbor (1-NN) algorithm for WSD is based on recent works showing the power of this approach in WSD task [58, 106].

#Documents	#Sentences	#Words	#Senses
26	1173	3663	4363

	Nouns	Verbs	Adj.	Adv
#Entities	2172	834	482	175
#Senses	2585	1051	536	191

Table 2.1: The statistics of the dataset.

2.4.1 WSD Experimental Setup

To test each embedding on the WSD task, we use the 1-NN algorithm and compare the disambiguated sense of each word with the ground truth annotations in the datasets. The details of the dataset we used for this experiment, the definition of accuracy, and the results are provided below.

Evaluation Datasets benchmarks

The dataset we use in this study to evaluate the quality of the embeddings in the WSD task is a unified WSD benchmark introduced by Raganato et al. [160]. This dataset is a combination of five standard WSD datasets. These sets are Senseval-2 [41], Senseval-3 [196], SemEval-7 [2], SemEval-13 [140], and SemEval-15 [131]. This unified benchmark includes 3663 words and 4363 senses in total. This dataset covers four parts of speech, including nouns, verbs, adjectives, and adverbs. The statistics of each part of speech in the dataset are shown in Table 2.1.

We use the 1-NN disambiguation approach, applying different contextual embeddings on the unified dataset and reporting the accuracy of each method. The accuracy is the percentage of words disambiguated correctly, based on the ground truth. The results of this experiment are shown in Table 2.2. The evaluation is based on the accuracy of the four considered embedding models. The high accuracy means the model is more accurate in disambiguating the ambiguous words as the ground truth data, picking the correct meaning (and also the correct word type). We use this evaluation and start analyzing the errors of each embedding approach.

Model	Senseval-2	Senseval-3	Semeval-7	Semeval-13	Semeval-15	All
BERT	77.1	73.2	66.1	71.5	74.4	73.8
LMMS	76.1	75.5	68.2	75.2	77.1	75.3
SensEmBERT	72.4	69.8	60.1	78.8	75.1	72.6
ARES	78.2	77.2	71.1	77.2	83.0	77.8

Table 2.2: The accuracy of 1-NN WSD evaluation framework on the unified dataset, using the four recent contextual embedding approaches as baselines.

2.4.2 Error Analysis by Part-of-Speech

To analyze the errors, first, we consider the disambiguation performance after using each embedding model on different word types covered by the dataset. In other words, we measure the frequency of mis-disambiguation in different parts of speech (POS). The accuracy of each embedding model using the 1-NN WSD approach on the dataset is shown in Table 2.3, which is categorized by the four parts of speech.

Model	Nouns	Verbs	Adjectives	Adverbs
BERT	76.2	62.9	79.7	85.5
LMMS	78.2	64.1	81.3	82.9
SensEmBERT	77.8	63.4	80.1	86.4
ARES	78.7	67.3	82.6	87.1

Table 2.3: The accuracy of the 1-NN WSD of the four considered embeddings as baselines. The dataset in this experiment is a concatenation of all five datasets, which is split by Part-of-Speech tags.

Among BERT, LLMS, SensEmBERT, and ARES, ARES has the higher accuracy in disambiguating the nouns with 78.7% accuracy, and BERT has the lowest accuracy of 76.2%, as shown in Table 2.3. On adjectives, between the four embedding methods, ARES is the most accurate one with 82.6% accuracy. LLMS is in the next place with 81.3% accuracy, and in the third place, we have SensEmBERT with 80.1% accuracy, and the last one is the BERT model with 79.7% accuracy. We can observe that the adverb has the highest disambiguation accuracy among all other word types in all four different embedding methods in Table 2.3. On the other hand, the verb is the type of word with the lowest disambiguation accuracy among all the four-word types. Verbs have the lowest frequency of correct disambiguation. In each embedding model, disambiguating the nouns is more accurate than verbs, which

confirms the result of a recent work by Loureiro et al., when the embedding model is BERT [106].

After observing the results of disambiguating different word types by the four various models, one question would be why almost all the models perform more accurately on disambiguating the nouns than verbs. There are multiple possible hypotheses. 1) Considering the coverage of the number of words in the dataset, in Table 2.1, this number differs for various word types. For example, since there are more nouns than verbs in the dataset, each model learns to disambiguate the nouns more accurately in comparison with verbs, or other word types. 2) The other important reason relates to the context. When the model disambiguates the ambiguous words based on nearest neighbors to the context, it is more likely that the model will choose the senses that are more similar to the type of context vector. In general, the number of nouns in English text (and every sentence) is higher than other word types, so the text's vector (sentence's vector) would be closer to the noun type. 3) One other possible reason is related to the ambiguity level. The ambiguity level is calculated as the total number of candidate senses, meaning the senses that share the surface form of the target word, divided by the number of the words [160]. Considering the ambiguity level of a specific word type as the number of all possible senses of that type divided by the number of words of that type can explain the behavior of these models. The ambiguity levels of different word types are relatively similar, by looking at Table 2.1, with verbs being slightly more ambiguous than nouns. This suggests that verbs may be more difficult for the model to disambiguate accurately, but this hypothesis is not yet convincing, as the ambiguity levels of different word types do not differ enough. 4) Finally, one other possible reason would be related to the POS tagger. The POS tagger would influence the accuracy of the disambiguator. To further investigate this hypothesis, we consider replacing the POS tagger and evaluate the results. The next section presents this investigation.

The Influence of POS Tagging on the WSD Task

To investigate the influence of the POS tagger, we try to answer this question: does an incorrect POS tagger influence the disambiguation accuracy? In other words, how much the model disambiguates the words incorrectly because of type confusion; for example, when disambiguating the word "play" in the sentence "The children's play was filled with laughter and joy", the model might get confused disambiguating it as a verb instead of a

noun. So, in the disambiguation task, some errors arise when the *word type* is mistaken. For quantitatively analyzing these errors, we consider *error rate* of each embedding approach, the Eq. 2.2, and we use a *confusion matrix*. The confusion matrix shows the misclassification rate of one specific part of speech inaccurately as the other parts of speech. To analyze the errors that arise due to incorrect word types, we consider running the same disambiguation experiments while changing the POS tagger.

A general overview of this procedure is provided in Fig. 2.1. We start this experiment by passing the input text into the POS tagger and then the embedding model. After getting the word vectors, we pass them into the 1-NN WSD algorithm to disambiguate the ambiguous words and compare the disambiguation results with the ground truth data. Based on this comparison, we can divide the errors into two categories: the words with incorrect disambiguation due to incorrect meaning and the others with incorrect disambiguation due to incorrect type. We focus on this second category and count the words in each word type. We then change the POS tagger and start this procedure to count the number of the disambiguated words in the four parts of the speech. By this counting, we will find how many words get disambiguated correctly after changing the POS tagger; it means how many of the wrong disambiguated words are now disambiguated correctly due to the POS tagger. First, we run the experiments with NLTK and then replace it with a deep-learning-based POS tagger, "en-core-web-md". This POS tagger is a pre-trained statistical model for English provided by the spaCy library [85]. This deep learning model is based on convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

$$\text{Error Rate (e)} = 1 - \text{Model Accuracy} \tag{2.2}$$

Considering the number of words in each word type category and finding the number of words that each model is disambiguating incorrectly in each word type, we report this analysis on ARES since it shows the best accuracy results in Table 2.3. We argue that errors made by these representation learning methods are similar regarding the word type. We present the rate of errors in each type in Table 2.4, in which the fifth column shows the percentage that each type is mis-disambiguated (Mis-D) because of the wrong sense (meaning), and the sixth column shows the percentage of times each type is mis-disambiguated because of the wrong type. The main observation extracted from Table 2.4 is that the representations produced by ARES fail to accurately capture the intended meanings

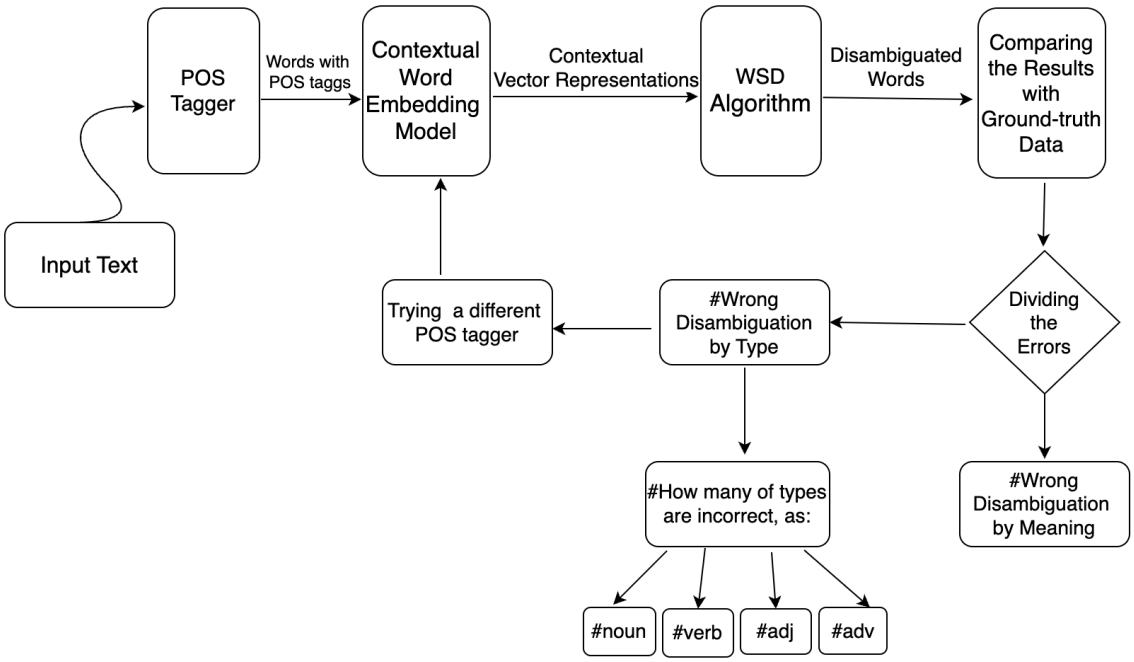


Figure 2.1: The procedure of analyzing the errors of the contextual word embedding models in the task of WSD and evaluating the effect of different POS taggers in these errors. These numbers are shown in Table 2.5 for nouns. For the other word types, please look at Fig 2.2, which shows the number of words that have been considered incorrectly as other word types.

of verb senses compared to other word types. This result is similar when the embeddings are extracted from BERT [106]. While the ARES representation is a context-based embedding method, it fails to address verb ambiguity 32.7% of the time.

Embedding Method	Type	Freq	ER	Mis-D by Sense	Mis-D by Type
ARES	Noun	2172	21.3%	16.7%	4.6%
	Verb	834	32.7%	25.4%	7.3%
	Adj.	482	17.4%	11.2%	6.2%
	Adv.	175	12.9%	8.5%	4.4%

Table 2.4: Error frequency analysis of the 1-NN WSD evaluation framework with ARES representations on the dataset, separated by type. Freq shows the frequency of the number of each word type in the dataset, ER is the Error Rate of each model on the word types, and Mis-D shows the percentage of the words that are mis-disambiguated in each word type.

To measure how the model is accurate regarding the word type after disambiguation, we count the number of nouns, verbs, adjectives, and adverbs for each word type after disambiguation. For example, when disambiguating the word type noun, we collect information regarding the model's correctness by considering the number of times when nouns are disambiguated as a noun. Also, we count how many of those nouns are disambiguated incorrectly as verbs, adjectives, or adverbs. This information for nouns is reported in Table 2.5. This table shows the total number of nouns in the dataset and how many have been disambiguated incorrectly. Between those that are disambiguated incorrectly, we report how many are disambiguated incorrectly because of the wrong meaning and how many are disambiguated incorrectly because of the wrong type. These nouns could be disambiguated incorrectly as verbs, adjectives, or adverbs. To this end, we report how many of these incorrect disambiguations by the type have been corrected after changing the POS tagger errors in this table, Table 2.5, using a deep learning-based POS tagger, "en-core-web-md".

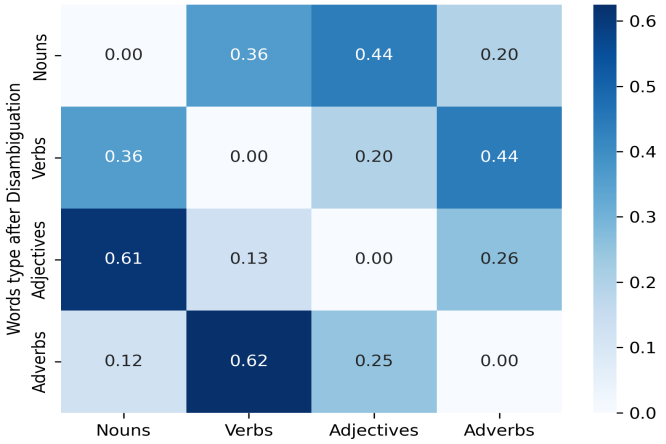
Similar to this experiment for nouns, we also run experiments for verbs, adjectives, and adverbs, representing the results in confusion matrices. The confusion matrices for disambiguating the four-word types using the ARES representation model are shown in Fig. 2.2.

#Noun = 2172		#D with incorrect Type=102		
		As Verb	As Adj	As Adv
#D-C = 1709	(I-S,C-T)	37	45	20
#D-C after fixing the POS tagger errors = 1725	361	31	38	17

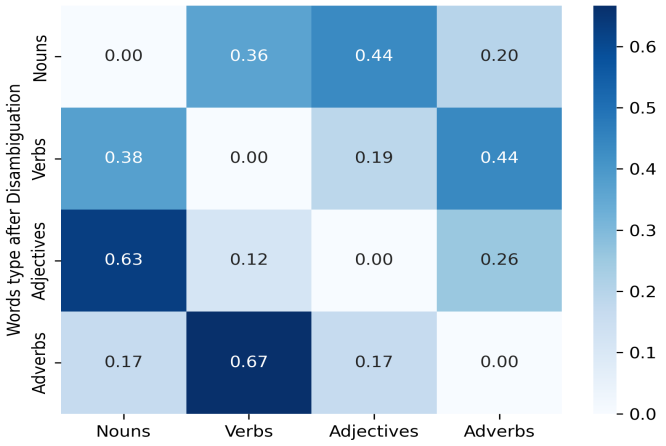
Table 2.5: Following the presented result in Table 2.4, here are the statistics of the number of Disambiguated (#D) words either Correctly (C) or Incorrectly (I). The lower row under the same column shows how many of those incorrect disambiguations got corrected after using the deep learning-based POS tagger and fixing the POS tagger errors by assigning the correct type of noun.

In this figure, each row of the matrix shows the percentage of incorrect disambiguation because of the wrong type plus the percentage of corrected disambiguated words after using another more accurate POS tagger. For example, between 463 incorrect disambiguated nouns using ARES, 361 are disambiguated incorrectly because of wrong meaning, as shown in Fig. 2.2, the left column of the first row. At the same time, their type is correct, which means they are labeled as nouns after disambiguation. Of the remaining 102 nouns, 37 are disambiguated as verbs, 45 as adjectives, and 20 as adverbs. These results indicate that part of this mis-disambiguation is because of the wrong word type, which shows the importance of the role of the POS tagger in the WSD task. This experiment helped to evaluate the POS tagger's role in our WSD task. Furthermore, this experiment indicated how much we can improve the results of WSD by fixing the errors that arise from the POS tagger, which is the wrong word type [130, 103]. Some of the recent common POS taggers that can be used for WSD include, but are not limited to, NLTK[5], and deep-learning and machine-learning based POS taggers [37, 114, 85]. Our result shows that about 1% of the errors have been corrected by using the deep-learning POS tagger, compared to when we use NLTK, when disambiguating the words. However, this improvement is not statistically significant. To evaluate how significant is the effect of changing the POS tagger in the final results of the WSD model, we need to run a statistically significant test, a t-test. The t-test determines if there is a statistically significant difference between the two methods. After conducting the two-sample t-test (independent samples t-test), with the significance level $\alpha = 0.5$, we conclude there is no significant difference between the means of these two sets (using deep-learning-based PoS tagger versus using NLTK) in each of the two experiments reported in Fig. 2.2.

[5]https://www.nltk.org/



(a) The confusion results whit NLTK as POS tagger



(b) The confusion results with en-core-web-md as POS tagger

Figure 2.2: The confusion matrix for the word types in the 1-NN disambiguation task, using ARES embeddings.

Using the same analysis, we evaluate the other three embedding approaches, BERT, LMMS, and SensEmBERT. These results show that, for example, when using BERT as the embedding approach in the WSD task, for 75% of nouns, the model is correct, meaning the word type after disambiguation is a noun. While at the same time, the model disambiguates 11% of the nouns as verbs. The BERT model also mis-disambiguates 1% of the nouns as adjectives and mis-disambiguates 0.04% of the nouns as adverbs. These experiments show that the ARES model is less confused when disambiguating the type verb than the other models. Another observation is that the verbs are misclassified as the wrong word type and mostly misclassified as nouns. Considering the calculation of the context vector at the disambiguation time, we can see that since most of the words in a sentence (and in the whole document in general) are nouns, the context vector is more like nouns (a noun-like vector). So, applying the 1-NN approach, it chooses the nearest words, which are the type nouns, because the type of the context vector is closer to the nouns than verbs or the other types.

These experiments demonstrate that LMMS and ARES effectively disambiguate the adjectives to the right type for adjectives and adverbs. At the same time, SensEmBERT and BERT are the two models that are less accurate when disambiguating the adjectives and mis-disambiguating them as nouns. The last type is the adverb, and the experimental results indicate that ARES is the model with less confusion when disambiguating adverbs compared to the other models. The SensEmBERT is the next model whose performance is near ARES regarding type confusion of adverbs. On the other hand, LMMS makes the mistake of type confusing for 0.11% of adverbs as verbs. BERT is next after LMMS of confusing adverbs as verbs, for 0.08% of the adverbs. These results indicate that while the contextual representations in disambiguation are important, we still need to pay attention to other factors, like the word's position in the sentence/document at the time of disambiguation.

The following example from Christopher Potts's course lecture shows a situation when the word type is the same but has various senses for the same word in different sentences.

a. The vase broke.
b. Dawn broke.
c. The news broke.
d. Sandy broke the world record.
e. Sandy broke the law.
f. The burglar broke into the house.

g. The newscaster broke into the movie broadcast.

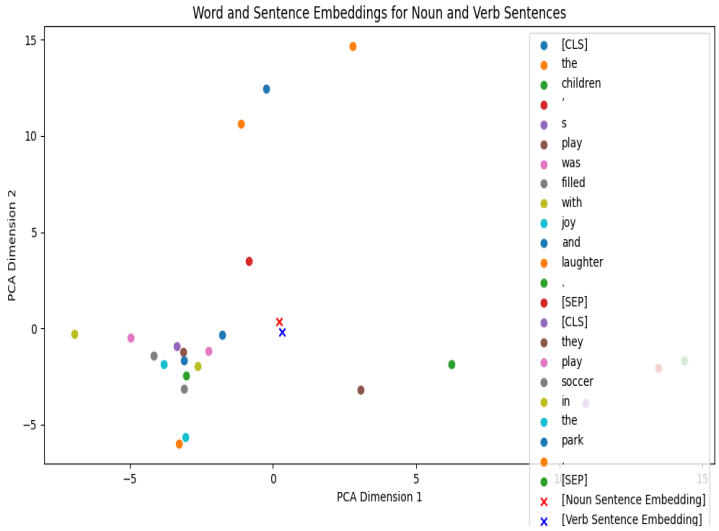
h. We broke even.

The word “broke” in all of these sentences has the verb type, but the meaning is different. This example shows the importance of context and the word’s position in the sentence. This observation indicates another interesting direction of positional encoding in the text ambiguity task.

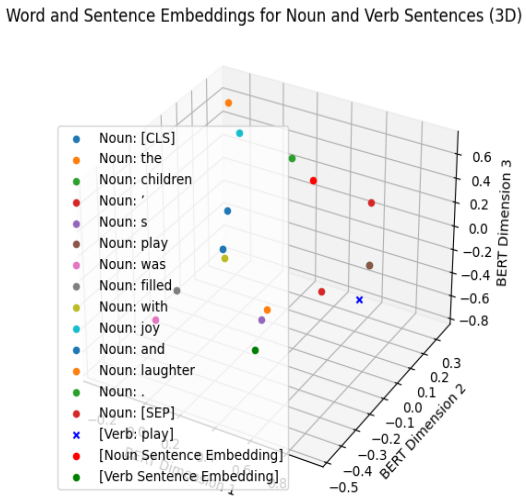
To better visualize the position of each word’s representation and the position of a sentence, we present the following scatterplot, Fig. 2.3. This plot represents the BERT vectors’ positions in 2D and 3D obtained by Principal Component Analysis (PCA) of the words in the sentence “The children’s play was filled with joy and laughter” and sentence “They play soccer in the park”. In the first sentence, the type of “play” is a noun, while in the second sentence, the “play” has a verb type. We show how close the position of vector representations of different meanings of one word (play) is in the space. Also, this plot,Fig. 2.3, includes the SBERT vector’s position representation of the sentence itself.

This experiment shows that the vector representation of “play” as a verb and as a noun might be too close to each other because the BERT-based models are trained on a large corpus of text without any explicit notion of parts of speech. Therefore, it is likely that the model has learned to represent different contexts and meanings of the word “play” using a very similar vector representation rather than an exact distinction between its noun and verb forms. This is a known limitation of BERT and similar models, which rarely capture fine-grained linguistic distinctions. However, it is essential to note that BERT was not specifically designed to encode syntactic or semantic roles, such as parts of speech, but to encode contextual word representations that capture the meaning of the word in its context.

Based on the above results, we need to consider good POS taggers when solving the WSD problem while focusing on other factors, such as the nature of the embedding approach we use in the WSD model. For this reason, we build a new embedding model that considers the context of the input text in its vectors more than previous approaches. Bringing more information from the context in the representation would help to choose the correct meaning for ambiguous words when applying the 1-NN approach. If the context vector representation is calculated more accurately, then the expectation is that the 1-NN approach would pick the more similar word type while picking the correct meaning.



(a) 2D plot of positions of embeddings (words and the sentence)



(b) 3D plot of positions of embeddings (words and the sentence)

Figure 2.3: Visualizing the positions of word embeddings and sentence embedding.

2.4.3 Results and Conclusion Based on Analyzing the Errors

In this section, we evaluated the accuracy of contextual pre-trained embedding approaches for the word sense disambiguation task. In this analysis, we considered the POS error rates of the embedding models and showed the analysis results for each considered embedding model. We further continued our quantitative analysis in different word types of the dataset. We analyzed the errors of each embedding model on parts of speech tags and used a confusion matrix for each word type. These confusion matrices show how the models disambiguate words incorrectly as a wrong word type. Among the different parts of speech, we observed that the verb is the most challenging type to disambiguate. Also, we evaluated some of the possible hypotheses that we could think of as potential reasons for these results. Our results indicate that the POS tagger does not significantly change the WSD results.

Conversely, the context vector embedding calculation is biased toward the type of words that are more frequent in the context. Considering how the context embedding vector is calculated, the WSD algorithm tends to pick noun word types for most of the words, including verbs. This biased embedding occurs because the context vector is more strongly associated with noun types than with other word types. Since in most English texts, the number of nouns is higher than other word types [113], the context vector would be closer to the noun type than other word types. So, building an embedding method in which the context information has been considered well in its representation is important. Using this contextual embedding helps to disambiguate different word types more accurately. We follow this direction in the next section, Section 2.5, and try building a new embedding model, considering the above experiments and observations. The other interesting possible direction includes evaluating the accuracy of the models in disambiguating the four-word types when the ambiguity level of all types is the same and training the embeddings on unbiased data.

2.5 New Proposed Representation Learning

We present a new approach to the representation learning problem based on our previous results in evaluating some of the most recent pre-trained language models. We first introduce our new proposed approach and its steps. Later, we show the effectiveness of this new

embedding approach in the task of WSD. We call our method "**C-KASE**", **C**ontextualized-**K**nowledge base **A**ware **S**ense **E**mbedding.

In our new C-KASE representations, we use Wikipedia, BabelNet [139], WordNet [122], SemCor [124] as the knowledge bases, and BERT [38], and SBERT [163] as the embedding models.

2.5.1 C-KASE

C-KASE is created by combining semantic and textual information; semantic information from hypernym and hyponym relational senses of BabelNet, textual information from the first paragraph of each sense's Wikipedia page, and the context from the input document text that includes the concept (the ambiguous word). Furthermore, C-KASE uses neural language models, i.e., BERT and SBERT, to create these representations. We divide our approach into the following steps.

- Pre-Processing Step: Semantic Context Retrieval

  In this step, the model retrieves all the relevant contextual information from BabelNet for every given ambiguous concept. We exploit the mapping between synsets and Wikipedia pages available in BabelNet. The reason for choosing BabelNet is due to its available hyponym and hypernym relations. It means that for each ambiguous mention, we gather all the concepts related to it through hyponym and hypernym connections in the BabelNet knowledge base. We consider R_s to be the set of extracted senses from BabelNet related to ambiguous mention s.

  When we gather all the related senses, we use the mapping between BabelNet and Wikipedia to consider the Wikipedia pages of those senses in R_s. For each Wikipedia page p_s, we consider the first paragraph of the page and compute its lexical vector representation by taking the average of the SBERT vector representation of the sentences in this first paragraph. These lexical representations are later used for computing the similarity score between p_s and $p_{s'}$, for each $s' \in R_s$ by using the weighted overlap measure [156], as we mentioned in Eq. 2.1. Pilehvar et al. show that the weighted overlap is preferred over the more common cosine similarity method to capture the senses' similarity more accurately. It performs better when comparing sparse vector representations [156]. Once we have scored all the (p_s, p'_s) pairs, we

create partitions of R_s, each comprising all the senses s' connected to s with the same relation r, where r can be one among hypernymy, and hyponymy. In our experiments, similar to the baseline [180], number of possible senses is set to 15, as this setting showed better performance over other choices.

- Creating the Embeddings

  In this step, we use BERT to extract the contextual embeddings of the given concepts (ambiguous words) from the input text. When considering the hyponymy and hypernymy relations based on BabelNet, we extract all the senses of the ambiguous concept from BabelNet and build the set of R_s. For each one of these senses, we use the link structure of Wikipedia and BabelNet to collect all the Wikipedia pages for each sense. Then, for each sense, we apply BERT to extract sense representations.

  Now, we build the final representation of each concept. We took the representation of mention (ambiguous word), $R(m)$, and the representation of each one of its senses from the previous step. We show the representations of each of the k senses of mention m with $R(s_i)$, where i varies from 1 to k. Our unique representations combine the mention's representation with sense representation, concatenating the two vector representations of $R(m)$ and $R(s_i)$. If mention m has k senses, C-KASE generates k different representations of $R(m, s_1)$, $R(m, s_2)$, ..., $R(m, s_k)$.

  In the first step, we took the representation of the input text paragraph, which contains the ambiguous mention. We show it by $R(PD)$, which stands for representation of the **P**aragraph of the input **D**ocument. In the first step, we also represent the first paragraph of the Wikipedia page, using SBERT. We represent it by $R(PW)$, which stands for representation of the first **P**aragraph of the **W**ikipedia page. Finally, we concatenate these two representations as $R(PD, PW)$. The dimension of this concatenated representation is also equal to the word representation using the L2-norm [142], making it possible to calculate their cosine similarities. To rank the most related senses to the context, we use the cosine similarity as follows:

$$\text{Sim}(m, s_i) = Cosine(R(m, s_i), R(PD, PW)) \text{ , } i = 1, ..., k \tag{2.3}$$

This ranking provides the most similar sense to the context for each mention. We use

the *BERT-base-cased* model in our experiments and SemCor. SemCor is a collection of sentences. The words in these sentences are annotated manually with meanings from WordNet. To this end, given a mention-sense pair (m, s), we collect all the sentences $c_1, ..., c_n$ where *m* appears tagged with s [153]. Then, we collect all the retrieved sentences, pass them into BERT, and extract the embeddings, BERT(c_1, m), ..., BERT(c_n, m). We get the average of the context vector concatenating with sense's gloss vector. It means getting the average of BERT(c_1, m), ..., BERT(c_n, m). Similar to the SensEmBERT approach, if a sense does not appear in SemCor, we only consider the vector with its sense gloss representation.

At the end of these steps, we have encoded the contextual information of the input text and the knowledge base information for each sense in one vector. This information has been collected from Wikipedia pages as well as the sense glosses from BabelNet. The k senses of each ambiguous mention are ranked in our representation based on their relevancy degree to the context, which is one of the novelties of our model. For this aim, we concatenate representations of the first step. The procedure for computing the C-KASE representations is shown in Fig. 2.4.

2.5.2 Experimental Setup

We present the settings of our evaluation of C-KASE in the English WSD task. The evaluation benchmark in this section is the same dataset we used for error analysis. This benchmark is the English WSD test set framework, constructed from five standard evaluation benchmark datasets[6], as is introduced in Section 2.4.1. This data includes Senseval-2 [42], Senseval-3 [197], SemEval-07 [157], SemEval-13 [138], SemEval-15 [131], shown as ALL, i.e., the concatenation of all the test sets [160]. All these datasets are WordNet-specific and mostly use SemCor [124] as their training set. The unified benchmark provides 7253 test instances for 4363 sense types, which covers 3663 word types across four parts of speech: nouns, verbs, adjectives, and adverbs.

[6] http://lcl.uniroma1.it/wsdeval/evaluation-data

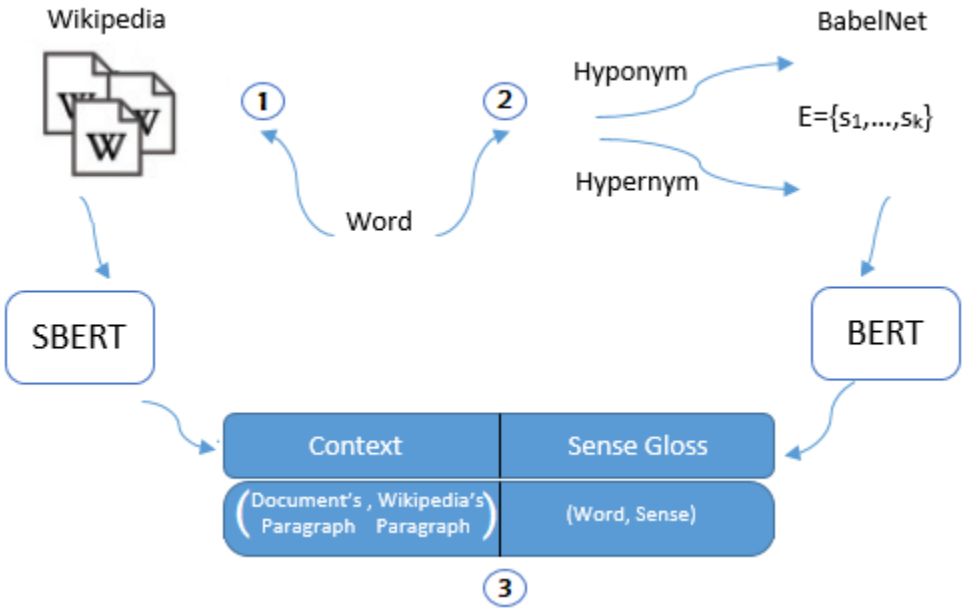


Figure 2.4: Demonstration of the C-KASE representation and its three components. Component 1) Collecting all Wikipedia pages for ambiguous words and possible meanings (senses), Component 2) Using hypernymy and hyponymy relations to extract all synsets for ambiguous words from Babelnet, Component 3) Concatenating (word, sense) representation for all senses from the second component with (Document's paragraph, Wikipedia's paragraph) representation from the first component as context.

Comparison Systems

We compare our representation with the same evaluated models on the English WSD task that we introduced in the previous section, employing the 1-nearest neighbor approach. LMMS is one of these systems that generates sense embeddings with complete coverage of WordNet and uses pre-trained ELMO and BERT models [104]. SensEmBERT is the next system that relies on different sources for building multilingual sense vectors [180]. These sources include Wikipedia, BabelNet, NASARI lexical vectors, and BERT. The next comparison system is ARES, a semi-supervised technique that builds sense embeddings [181]. To our knowledge, ARES is the most recent contextual word embedding system at the time of building our model. In our evaluations, we also consider BERT as one of the comparison systems since it is at the core of all the considered methods [59]. The results comparing the accuracy of our method and the baseline, ARES, are presented in the following tables, Table 2.6 and Table 2.7.

Model	Senseval-2	Senseval-3	Semeval-7	Semeval-13	Semeval-15	All
ARES	78.2	77.2	71.1	77.2	**83.0**	77.8
C-KASE	**79.6**	**78.5**	**74.6**	**79.3**	**83.0**	**78.9**

Table 2.6: The accuracy of 1-NN WSD evaluation framework on the unified dataset, using our model and comparing it with the winner of the baselines, ARES. The result of ARES is the same last row of Table 2.2, above. C-KASE is our new proposed contextual embedding model, introduced in this thesis (Section 2.5). The higher accuracies are highlighted.

Model	Nouns	Verbs	Adjectives	Adverbs
ARES	78.7	67.3	82.6	87.1
C-KASE	**79.6**	**69.6**	**85.2**	**89.3**

Table 2.7: The accuracy of the 1-NN WSD of our model, C-KASE, and ARES, which is the best one between the baselines, are taken from the last row of Table 2.3. C-KASE is our new proposed contextual embedding model, introduced in this thesis, Section 2.5. The dataset in this experiment is a concatenation of all the five datasets, which is split by Part-of-Speech tags. The higher accuracies are highlighted.

Ablation Study

When creating our new representations, we first concatenated the BERT representations of ambiguous words and the BERT representation of one of the senses. Second, we concatenated the SBERT representations of input text and the Wikipedia page's text. Then, we compared the similarity between these two concatenated vectors to find the most similar sense based on the information from the knowledge base and the input text. One question that one might ask would be the impact of each component of this representation in the final disambiguation results. For this aim, we compare the results of the previous method with those obtained when only using the BERT representation of the ambiguous words, and compare it with the SBERT representation of the Wikipedia page's text, showing it with C-KASE'.

In this experiment, all the settings are as before, with the only difference in the dimension of the representation vectors. In this experiment, since we do not concatenate the two representations, the dimension of the final representation vector is the same as the dimension of the vectors extracted from SBERT, while in the previous experiment (the original idea), it was doubled.

The results of evaluating this version of our C-KASE representations are reported in Table 2.8 and Table 2.9. These two tables are similar to the reported results in Tables 2.6 and 2.7, while we only provide the last row of those two tables here, which is C-KASE. The plot visualizations of these two tables are also depicted in Fig. 2.5 and Fig. 2.6.

Model	Senseval-2	Senseval-3	Semeval-7	Semeval-13	Semeval-15	All
C-KASE'	76.3± 0.42	73.1± 0.15	67.1± 0.25	71.2± 0.22	73.6± 0.33	72.8± 0.21
C-KASE	**79.6± 0.11**	**78.5± 0.23**	**74.6± 0.17**	**79.3± 0.26**	**82.9± 0.15**	**78.9± 0.12**

Table 2.8: The accuracy of 1-NN WSD evaluation framework on the unified dataset, using our C-KASE representation, introduced in Section 2.5, and C-KASE'. The C-KASE' is the modified version of C-KASE that compares the representation of senses (extracted from the Wikipedia page of the sense) with a representation of input text. The higher accuracies in each word type are highlighted in bold.

To have a better perspective of how effective is the original version of C-KASE versus the modified version C-KASE', we run a statistically significant test, a t-test. As we know, the

Model	Nouns	Verbs	Adjectives	Adverbs
C-KASE'	75.7± 0.23	61.9± 0.14	79.3± 0.21	84.7± 0.12
C-KASE	**79.6± 0.32**	**69.6± 0.22**	**85.2± 0.15**	**89.3± 0.34**

Table 2.9: The accuracy of the 1-NN WSD using the two versions of C-KASE. The C-KASE is our method introduced in Section 2.5, and C-KASE' is the modified version. This experiment indicates the effectiveness of considering contextual information from both the knowledge base and the input text during disambiguation. The higher accuracies in each word type are highlighted in bold.

t-test determines if the two methods have a statistically significant difference. We conduct two independent sample t-tests with $\alpha = 0.05$. The test shows the original idea produced significant improvements compared to the modified version. This experiment indicates the effectiveness of contextual information from the knowledge base and the contextual information from the input text. Considering the spatial space, this contextual information in the representations makes the vectors closer to the correct meaning. By incorporating contextual information, word representations can better capture the nuances of meaning and relationships between words, improving the ability of word representations to accurately represent the meaning of words in different contexts.

2.6 Results

We compare the effectiveness of C-KASE representation with the existing state-of-the-art models on the standard WSD benchmarks. We report the results of C-KASE in Table 2.2 and compare them against those obtained from other state-of-the-art embedding approaches, as shown in Fig. 2.7. All performance metrics are reported using accuracy.

The overall performance of all the models on the All dataset is higher than 70%, which is a good accuracy for the task of WSD, considering the literature. Our model, C-KASE, has a performance of 79% on the All dataset. ARES is the model with better performance on each dataset between the baselines. After ARES, LMMS has higher accuracies on the disambiguation task. This model, LLMS, has an accuracy of 75.3% on the All dataset, and 76.1% on Senseval-2. BERT accuracy on Senseval-2 is higher than LMMS with 77.1% accuracy. SensEmBERT performance on SensEval-13 is higher even than other baselines. The reason is because of the number of different word types in these datasets. The number of nouns in SensEval-13 is higher than in other datasets, and as we mentioned above, the

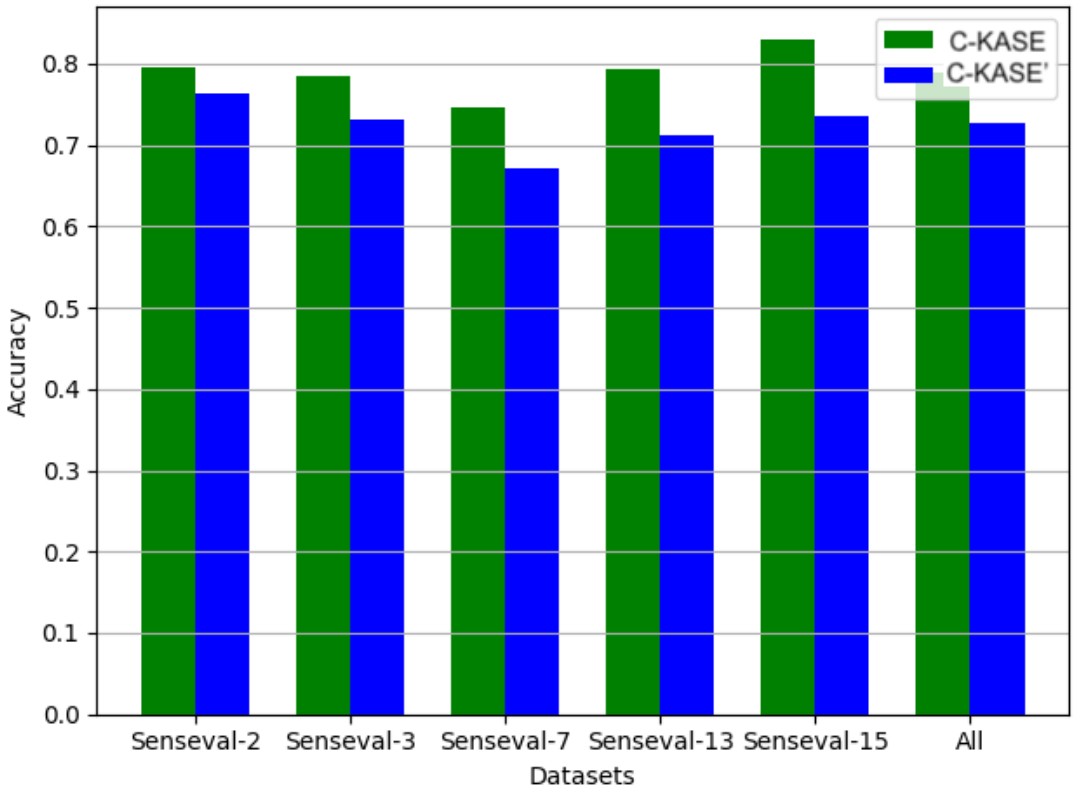


Figure 2.5: The accuracy comparison of the C-KASE model and the modified version of C-KASE, on our WSD datasets. The green represents the results of the original C-KASE method, and the blue represents the modified version, which we refer to as C-KASE'.

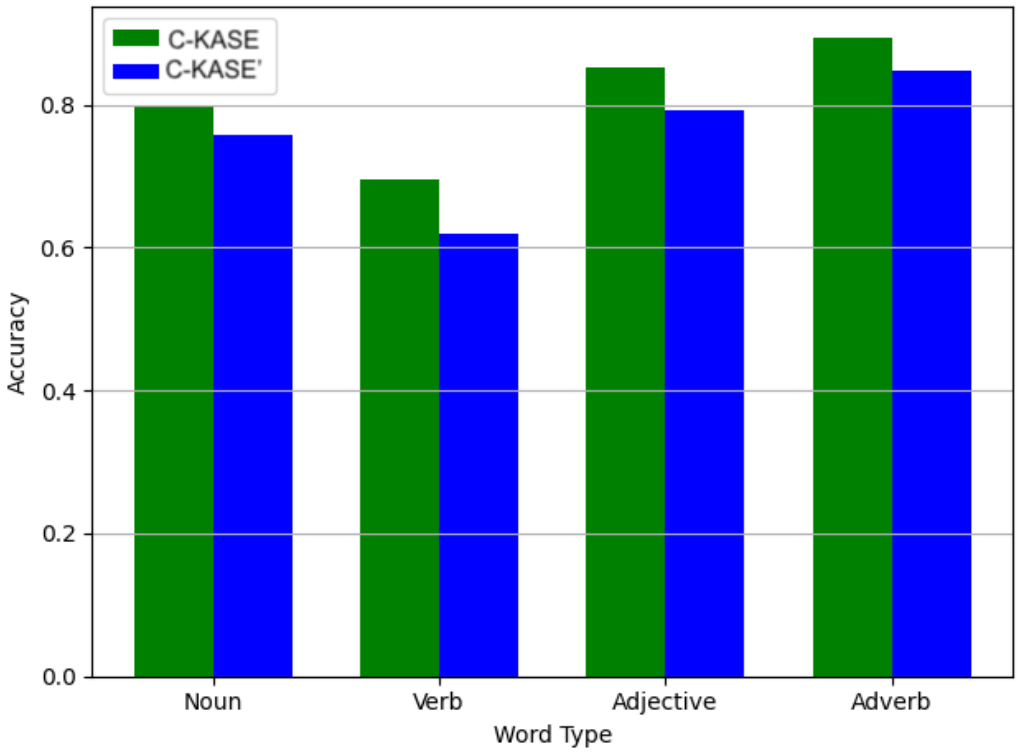


Figure 2.6: The accuracy comparison of C-KASE and its modified version on the four-word types of nouns, verbs, adjectives, and adverbs. The green represents the results of the original C-KASE method, and the blue represents the modified version, which we refer to as C-KASE'.

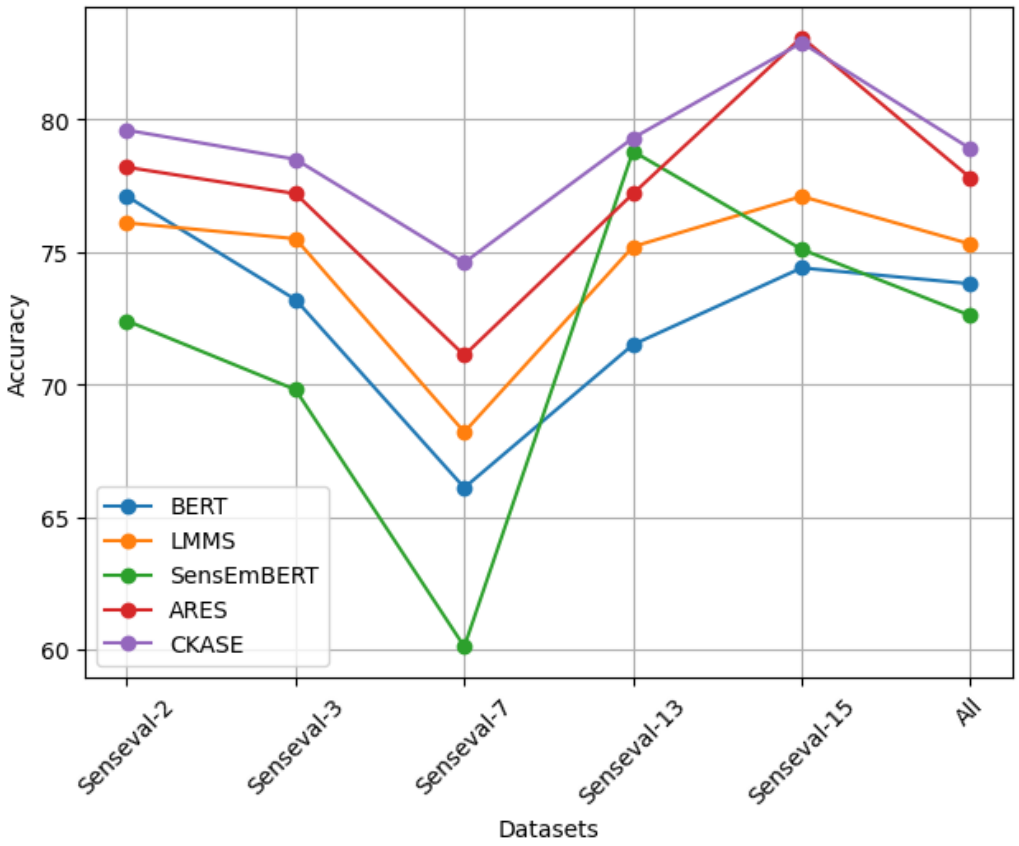


Figure 2.7: Accuracy comparison of 1-NN WSD evaluation framework on the unified dataset, using BERT, LMMS, SensEmBERT, ARES as baselines and our method, C-KASE.

disambiguator model tends to pick the senses closer to the context vector. Since most of the words in an English text are nouns, the type of the context vector is closer to noun type senses.

2.6.1 Discussion

The intuition behind C-KASE lies in incorporating the right contextual information. This contextual information, from the input text and the knowledge base, helps to better capture the nuances of word meanings in different contexts. Below is an explanation of why we designed C-KASE in this way and discuss the potential for performance improvement.

- Contextualized Representation: C-KASE uses contextual information from two types of sources. (1) Neural language models like BERT and SBERT to create representations that capture the context of words in a sentence. This context coming from the input text is one part of the contextual information that can be used at the time of disambiguating the ambiguous concepts. (2) The context from knowledge bases. C-KASE incorporates semantic information from knowledge bases such as BabelNet, WordNet, and SemCor, which contain hypernym and hyponym relations. This information enriches the representations with structured semantic knowledge about words, which can improve word sense disambiguation results.

- Weighted Overlap Similarity Measure: C-KASE uses the weighted overlap measure to compute the similarity score between word senses based on the first paragraph of their Wikipedia pages. This measure is preferred over simple cosine similarity for sparse vector representations, as discussed above, potentially improving the representation quality. Using a more enhanced representation makes the model more likely to produce accurate word sense disambiguation results.

C-KASE's good performance on the WSD task suggests it is a competitive alternative to previous context-based representation learning models. The evaluation results show that the idea of including context from the input text (the paragraph including the ambiguous word) and the knowledge base (the first Wikipedia paragraph) creating this C-KASE representation has improved the results of the lexical ambiguity task. It is a good indicator of the dependency of the WSD task on the representation learning component that is aware of the context and the information extracted from the knowledge bases. However, it is essential to critically

evaluate C-KASE's performance on various tasks, languages, and datasets to fully validate its superiority over other embeddings; I seek to follow these steps in my next research works.

The other evaluation we conducted is evaluating the effectiveness of each representation across different parts of speech: nouns, verbs, adjectives, and adverbs, using the All dataset, in which Table 2.1 provides the number of instances for each category. The accuracy of the 1-NN WSD for each contextual word embedding is summarized in Table 2.3 and visually depicted in Fig. 2.8. Upon analyzing the results, we observed that the performance of all the baseline models and our proposed model, C-KASE, on verbs, is comparatively lower than on other word types. The accuracy ranges from 62.9% for BERT to 69.6% for C-KASE. Regarding nouns, the models exhibit slightly lower performance when compared to adjectives and adverbs. BERT achieves an accuracy of 76.2% on nouns, while LMMS, SensEmBERT, ARES, and C-KASE achieve 78.2%, 77.8%, 78.7%, and 79.6% accuracy, respectively. On the other hand, for adjectives, the models demonstrate varying degrees of accuracy, ranging from 79.7% for BERT to an impressive 85.2% for C-KASE. Remarkably, the type with the highest accuracy across all models is the adverb, with 85% accuracy. These findings shed light on the strengths and weaknesses of each representation for disambiguating different parts of speech. Notably, C-KASE outperforms other models on adjectives, showcasing its potential for effective word sense disambiguation.

2.7 Contextual Representation Learning in Biomedical Word Sense Disambiguation

In the previous section, we have shown the effectiveness of our C-KASE representation learning approach on the general English text. In this section, we apply a similar idea to generate contextual word representations in the domain of Biomedical text. This is, in fact, a use-case of C-KASE representation learning in the biomedical domain, which we call "BioCBERT". The details of generating this representation learning and experiments are described in the following.

2.7.1 BioCBERT

BioCBERT stands for Biomedical Contextual BERT embedding. It is a contextual pre-trained representation learning model in Biomedical text. Like C-KASE, the BioCBERT

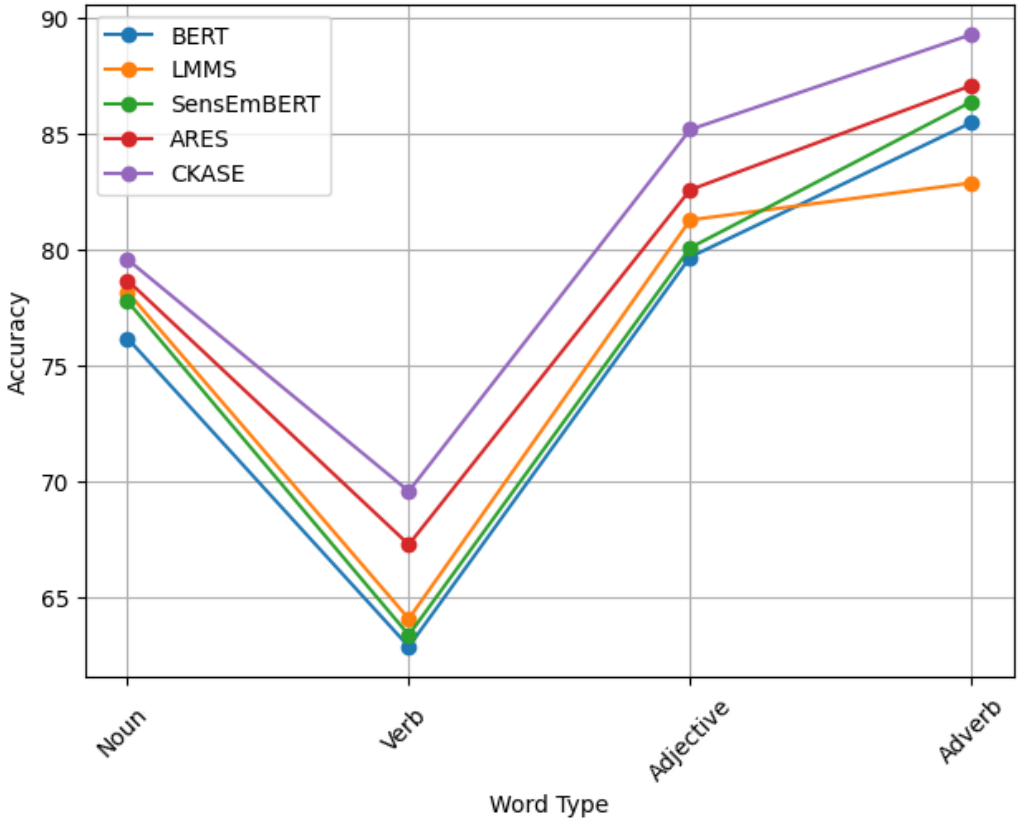


Figure 2.8: Comparing the accuracy of the 1-NN WSD of the four considered embeddings as baselines, and our method, C-KASE. The dataset in this experiment is a concatenation of all five datasets, which is split by Part-of-Speech tags.

is created by combining semantic and textual information from the knowledge base and the input text. BioCBERT uses neural language models, i.e., BioBERT and SBERT [163]. BioBERT is a pre-trained contextual language representation model [93], based on BERT, and is trained on different combinations of general and biomedical domain corpora. BioCBERT leverages additional knowledge sources like Wikipedia and UMLS during its development. [7]. BioCBERT follows similar steps as the original idea of C-KASE, including semantic context retrieval and creating the word and sense embeddings.

Semantic Context Retrieval

This first component aims to collect contextual information (similar to the first step of C-KASE) from the knowledge base, enhancing the representations. For each ambiguous word in the input text, we create a set including candidate senses for the word from UMLS. Like SensEmBERT [180], this procedure aims to collect relevant contextual information from the knowledge base for each given concept in the semantic network. Similar to the first step of C-KASE, we exploit the mapping between synsets and Wikipedia pages available in UMLS and its taxonomic structure to collect textual information relevant to a target synset s. For each synset s, we collect all the connected concepts to s from the UMLS. We show this set of related synsets to s by R_s. In this work, similar to the C-KASE idea, for each s in R_s, we get the Wikipedia page p_s, using the link structure of UMLS and Wikipedia. Then, we consider the first opening paragraph of the page and compute its lexical vector by taking the average of the SBERT vector representation of the sentences in this first paragraph. These lexical representations are later used for finding the similarity score between p_s and $p_{s'}$, for each $s' \in R_s$ by using the weighted overlap measure [156], which is defined by Pilehvar et al. as in Eq. 2.1.

Next, we use BioBERT to extract the representation of the given ambiguous words from the input text. For each ambiguous word (mention) of the input and the extracted senses from UMLS in R_s, we take the BioBERT representation. Now, we build the final representation of each mention by concatenating the representation of a mention with its sense's representation and then rank the representations based on their relevancy to the context, which here is the concatenated vector of Wikipedia (the first paragraph) and the input document text (the paragraph including ambiguous word). With this ranking based on

[7] https://www.nlm.nih.gov/research/umls/index.html

the similarity to the context, we can find the most related sense of the ambiguous mention to the context.

2.7.2 Experimental Setup

This section provides information on the settings of our BioCBERT evaluation in the WSD task in biomedical text. To test each embedding on the WSD task, similar to the previous experiment, we use the 1-NN algorithm. The nearest neighbors strategy is effective with pre-trained language models [106]. Then, we compare the disambiguated sense of each word with the ground truth annotations in the datasets. For evaluating the WSD approaches on medical text, we use the MSH dataset[8]. This dataset includes 37888 instances for 203 ambiguous terms [152].

2.7.3 Results

For evaluation, we compare BioCBERT against other related recent methods in the biomedical domain, including BioBERT [93], BioGraph [40], and deepBio [152]. We report the results in Table 2.10.

Model	Accuracy
BioBERT	83.4
BioGraph	71.52
deepBio	92.16
BioCBERT	94.71

Table 2.10: The accuracy results of 1-NN WSD evaluation on the MSH WSD dataset, using the most recent pre-trained embedding approaches on biomedical text, and compare those embeddings with our new proposed embedding, presented in the last row.

All four experimented methods achieved good performance on the Word Sense Disambiguation (WSD) task. BioGraph has an accuracy of 71.52% on disambiguating the words, and BioBERT has an accuracy of 83.4%. As we can see, the difference between BioBERT and BERT (evaluated in the previous section) is considerable, and the reason is obvious as BioBERT is pre-trained on domain-specific biomedical and clinical text, including PubMed

[8] https://lhncbc.nlm.nih.gov/ii/areas/WSD/collaboration.html

abstracts and PubMed full-text articles (14 million documents). BioBERT's vocabulary is specific to the biomedical domain, including medical terms, abbreviations, and specialized jargon used in the medical literature. The next model is deepBio, which shows a good performance of 92.16%, which is the closest to the performance of our BioCBERT model and has an accuracy of 94.71%. We can see from the results presented in this table that our model works well compared to other models. The result indicates that as the considered task is WSD in biomedical text, involving the right information from the knowledge base and context of the input text, as well as adaptability to specific domains, helped to improve the results, which is the main advantage of our approach. In our model, we observed the right information includes the first paragraph of Wikipedia (the knowledge base) and the focused paragraph (the paragraph including the ambiguous word) from the input text.

2.8 Conclusion

In this chapter, first, we evaluated the performance of some of the recent contextual embedding models in the task of WSD, using the 1-NN approach, and analyzed their errors. Second, based on the analyzed errors, we presented C-KASE, a novel embedding approach for creating sense embeddings considering context from a knowledge base and the context of the input document text. We showed that this context-rich representation is beneficial for lexical ambiguity in English. The results of experiments in the WSD task demonstrate the effectiveness of C-KASE representations compared to other state-of-the-art methods, despite relying on English data only. Furthermore, the results across different datasets showed how good the quality of our representations is. We further tested our embeddings using the data split into four parts of speech, including nouns, verbs, adjectives, and adverbs.

We applied the same idea of this new representation learning model to include context and knowledge base information in the biomedical domain and call this new representation learning BioCBERT. While using this idea in the biomedical domain, we still focused on the ambiguity task. By evaluating our representation model on the WSD task and comparing it with other context-based embedding approaches in the same task, we observed the effectiveness of our proposed representation. This work showed how context could play an essential role in the final results of the word sense disambiguation systems. The integrated information from the input text and knowledge base leads to a better choice of the disambiguation system as the correct meaning for the ambiguous words.

We have multiple directions for future work. (1) Extending our approach to other languages, considering the WSD task. By making WSD algorithms more accessible to people who speak other languages, we can help break down language barriers and facilitate communication between people from different countries. In this direction, we need access to a lexicon or dictionary for the language we are interested in and a corpus of text in that language. The corpus should represent the different senses of the words in the language and be large enough to train our WSD model. (2) Fine-tuning the pre-trained language models used in C-KASE with unbiased data. As the results of our error analysis on general English text show, the effectiveness of the contextual embeddings in WSD varies on different word types. One possible reason for this issue is the requirement for an equal distribution of instances within each word category in the dataset. We conjecture that if we can have a more balanced dataset regarding the number of instances in each word category, the algorithm will have a more fair opportunity to disambiguate words of each category correctly. This direction would prevent the results from being biased toward specific word types.

Chapter 3

Disambiguation in Wikification

Wikification is a method to automatically enrich a text with links to Wikipedia as a knowledge base. One step in Wikification is detecting ambiguous mentions, and another is disambiguating those mentions.

In this chapter, we present our work on the mention disambiguation problem. Some state-of-the-art disambiguation approaches [173] have divided long input document text into non-overlapping windows because of contextual relevancy and effectiveness in processing long text. Later, for each ambiguous mention, they pick the most similar sense to the chosen meaning of the key-entity (a word that helps to disambiguate other words of the text). Partitioning the input into disjoint windows means the most appropriate key-entity to disambiguate a given mention may be in an adjacent window. The disjoint windows negatively affect the accuracy of these methods. This chapter presents CACW (Context-Aware Concept Wikifier), a knowledge-based approach to finding the correct meaning for ambiguous mentions in the document. CACW incorporates two algorithms; the first uses co-occurring mentions in consecutive windows to augment the contextual information to find the correct sense. The second algorithm ranks senses based on their context relevancy. We compare the results of these two proposed algorithms with the other recent wikifiers and show how our methods improve the accuracy of the final results. We also define a new metric for disambiguation to measure the coherence of the whole text document. Comparing our approach with state-of-the-art methods shows the effectiveness of our method in terms of text coherence in the English Wikification task. We observed 10-20 percent improvement in the performance compared to the state-of-the-art techniques[1].

[1] Part of this chapter is published in [167].

3.1 Introduction

The task of "**Text Disambiguation**" is to identify mentions[2] of entities that have multiple meanings and link them to the correct relevant entry in an external knowledge base. This correct relevant entry is the correct meaning of the mention in that knowledge base, based on the input text [1, 164, 127]. Automated text disambiguation has become an important topic due to the vast growth in Natural Language Processing (NLP) applications [203, 212]. In computational linguistics, text disambiguation is an open problem of NLP, which concerns the automatic identification of a text's correct meaning. This task is related to the word-sense disambiguation (WSD) task, which is a narrower task that specifically focuses on resolving the ambiguity of individual words, relies on a predefined sense inventory, and aims to solve the ambiguity of words in a context. One benefit of text disambiguation is enhancing text readability and unambiguousness by inserting connections between the text and an external knowledge base, like Wikipedia, WordNet, BabelNet, and other knowledge bases. When Wikipedia is the knowledge base in text disambiguation task– which is the most popular among online encyclopedias [159]– the task is called the Wikification problem.

The problem of Wikification is closely related to other core NLP tasks, such as named-entity recognition (NER) [210, 212, 28, 200]. Wikification, in particular, is the task of associating a word in context with the most suitable meaning from the predefined sense inventory of Wikipedia. Named-entity recognition involves identifying certain occurrences of noun phrases as belonging to particular categories of named entities [64, 143, 116]. These expressions refer to names, including person, organization, location names, and numeric expressions, including time, date, money, and percent expressions [119, 133, 96].

The knowledge base is a centralized repository for information, like WordNet and Wikipedia [87]. Therefore, the performance and details of each WSD method are highly dependent on the knowledge base to link to. Knowledge bases are different in nature; for example, WordNet is a lexical graph database of semantic relations (e.g., synonyms, hyponyms, and meronyms) between words, while Wikipedia is a hyperlink-based graph between encyclopedia entries [134, 4].

Wikipedia is a free online encyclopedia that has grown into one of the largest online repositories of encyclopedic knowledge, with millions of articles available in various languages [22]. One of the essential attributes of Wikipedia is the abundance of links embedded

[2] A mention can be one or more tokens

in the body of each article, connecting the most important terms to other pages. Wikipedia contains one page per entity. Therefore, more detailed and related information can be found by following the links on such a page. Links make Wikipedia an excellent information hub, and representing the sense of a mention by linking to the corresponding Wikipedia page is a great way to represent disambiguation results to the user.

The content of Wikipedia pages could identify the similarity between entities and the structure of Wikipedia, as first presented by Mihalcea [118]. Links between Wikipedia pages represent relationships between the entities explained on those pages. Moreover, each page has a short introductory paragraph that sufficiently defines the entity. Both links and the introductory paragraphs can identify the similarity between entities (necessary for coherence-based methods) based on common neighborhoods in the Wikipedia graph and common terms in the introductory paragraphs. Our work uses Wikipedia as a source for automatic keyword extraction and entity disambiguation and is based on the 2020-11-01 English version of Wikipedia[3].

In the Wikification task, two steps are entity recognition and entity linking. The entity recognition step is also known under the names of *Spotting* or *mention detection*. It means identifying the terms that should be wikified. These are the mentions with multiple meanings in the knowledge base that need to be wikified. The other step is *Entity Linking* or *Disambiguation to Wikipedia*. In entity linking, we try to find the most related Wikipedia page [55]. These two steps are entirely separate steps. First, the spotter operates on the text to extract all ambiguous mentions and assigns all potential entity candidates for each mention. The entity linker disambiguates the candidate entities by selecting the most probable sense entity for each mention [193]. Our focus is on the second step, and we use the output of the recent spotting system, Wikisim [173] as the input to our algorithm (more details in Section 3.3).

A human reader can identify the correct meaning of each word based on the context in which the word is used. Computational methods try to mimic this approach [192, 46]. These methods often represent their output by linking each word occurrence to represent the chosen sense explicitly [217].

In this chapter, we categorized previous works on text disambiguation problem into two groups: The machine learning-based approaches and the knowledge-based approaches [181]. In the machine learning-based approach, systems are trained to perform the task [202,

[3] https://dumps.wikimedia.org/enwiki/

172]. The knowledge-based approach requires external lexical sources such as Wikipedia, WordNet [123], a dictionary, or a thesaurus. The machine learning-based approaches mainly focus on achieving maximal precision or recall and have run-time and space requirement drawbacks during classifier training [23]. So, knowledge-based Wikification methods still have advantages to study. One of the knowledge-based methods is the coherence-based Wikification method [39]. In the coherence-based approach, one important factor is the coherence of the whole text after disambiguation, and not the coherence of only a sentence. This coherence factor of the whole text might change by considering the coherence of each sentence or paragraph in other approaches. It is a significant challenge to perform Wikification accurately and fast enough to process long text documents [112, 191].

One observation has been used to solve this ambiguity problem in long documents efficiently. The observation says, "In a sufficiently short text segment, there is a word, named *key-entity*, which can be assumed to represent the central concept of the text and can help to disambiguate other terms in that text" [91]. The coherence-based approach using such a key-entity models the relatedness between the senses and the *key-entity* in disjoint windows of the text to speed up the disambiguation process. It disambiguates every word so that the total pairwise relatedness of all chosen word senses and key-entity of the same window is maximized [154, 174]. This method is still computationally expensive, and run-time performance is considered a secondary issue in most existing Wikification methods [193, 154].

Our work aims to implement a method that performs entity disambiguation and links to Wikipedia's most relevant pages. We demonstrate that using overlapping windows instead of disjoint windows in the baseline key-entity-based approach of Wikisim [173] leads to a new Wikification system with increased accuracy while increasing the computational cost only slightly. We named this first overlapping windows algorithm 1, OWCW (Overlapping-Windows Concept Wikifier), and further improved it by ranking senses based on similarities to the context in algorithm 2, as CACW (Context-Aware Concept Wikifier). All the code for this project is publicly available [4].

CACW is a knowledge-based approach to detect the correct sense for ambiguous entities in a text-based context and link them to Wikipedia's most relevant pages. We refer to Wikipedia as our *knowledge base* in this study. Our experimental evaluation, conducted

[4]https://github.com/mozhgans/wikification

on various datasets, demonstrated a higher level of accuracy using our method compared to previous Wikification methods. A careful inspection of the word senses selected by our method revealed that our method corrects most of the disambiguation errors made by the baseline approach. These errors arise from partitioning the input into disjoint windows in the baseline method. The baseline method did not disambiguate any entity correctly that was incorrectly disambiguated by our method.

The contributions of this work are summarized as follows:

- A literature review of the Wikification methods.
- Proposing OWCW, an algorithm for the disambiguation task, which uses the idea of overlapping windows.
- Proposing CACW, an improved version of the first proposed algorithm, ranks sense candidates based on context (the input text).
- Running experiments on short and long text to evaluate the performance of algorithms.
- Representing comparisons of run-time and performance of our approach and previous baseline approaches.

3.2 Related Work

Many NLP systems for the word sense disambiguation task rely on lexical knowledge bases (LKB). These knowledge bases vary significantly in structure, size, and subject, making them more appropriate for specific domains. For instance, WordNet was used for *synonym exploration* [57, 132], and the sophisticated Unified Medical Language System (UMLS) ontology was used for *medical text disambiguation* [83, 109]. On the other hand, disambiguation based on Wikipedia has been demonstrated to be comparable in terms of coverage to domain-specific ontology [216, 208] since it has broad coverage, with documents about entities in a variety of domains [112]. Moreover, Wikipedia has unique advantages over the majority of other knowledge bases [229, 112]. For example, mentions of entities in Wikipedia articles often link to the relevant Wikipedia pages, thus providing labeled examples of entity mentions and associated anchor texts (clickable text within a hyperlink) in various contexts, which could be used for supervised learning in Wikification.

Using any knowledge base for text disambiguation requires an "entity linker". When UMLS is used as the knowledge base, MetaMap is widely accepted as the entity linker [78, 5]. In the case of Wikipedia, the entity linker in the literature is referred to as **Wikifier**. In most studies, a Wikifier uses one specific feature from the set of features, including the context around the entity mention [220], and some data-driven statistics regarding the mention-entity relation, such as *commonness* or *prior probabilities* [162]. The most famous example is the **semantic coherence measure**. This feature is established based on the assumption that words in a given *neighborhood* (i.e., a segment of the text) will tend to share a common topic. Examples of widely accepted Wikifiers include Wikify! [118], Wikipedia Miner [125], TagME [45], and GLOW [162].

Wikify! is a Wikification method that disambiguates and ranks candidates to indicate the most valuable ones to the user in terms of meaning coherence. To choose the most appropriate sense, Wikify! uses the Lesk algorithm [94]. The Lesk algorithm identifies the most likely meaning for an ambiguous word based on the *contextual overlap* between the content of the Wikipedia pages corresponding to the candidate senses and the local context of the ambiguous word. This can be expected to maximize semantic coherence between the chosen meaning and the entity's context. In Wikify!, the Lesk algorithm counts the number of common words between the text paragraph where the ambiguous word occurs and the Wikipedia page corresponding to each candidate sense to choose the sense with the most common words. Wikify! is the first widely accepted Wikifier. However, it has already been outperformed by more recent Wikifiers, such as large-scale named entity disambiguation [35].

Large-scale named entity disambiguation aims to find the best match between the contextual information of each candidate sense and the context of the text [35]. The contextual information for each candidate sense is derived from a combination of features extracted from its corresponding Wikipedia page. One particular feature is the set of incoming Wikipedia links associated with each candidate sense. However, relying solely on these incoming links can introduce a challenge. It is possible for irrelevant entities to be included in this set simply because they share a significant number of common words with the input text. Consequently, an irrelevant candidate entity might be mistakenly chosen. To overcome this limitation, in our proposed method, we do not consider the incoming links but instead focus on measuring the similarity between candidate senses and maximizing the

overall coherence of the text. By prioritizing the similarity and coherence of the entire text, our method avoids the drawback associated with large-scale named entity disambiguation that arises from relying on potentially misleading incoming links.

The next Wikifier is Wikipedia Miner [125], outperforming Wikify! and the large-scale named entity methods [118, 35] introduced above. The entity disambiguation approach of Wikipedia Miner relies on the graph structure of Wikipedia. This structure performs disambiguation based on two concepts: *commonness* and *relatedness*. The commonness of a sense is the number of times that the sense is used as a destination in Wikipedia [144]. Hence, commonness is sometimes referred to as prior probability. The relatedness of a candidate sense is its similarity to the context, based on the commonness. Their approach aims to balance the commonness of a sense with its relatedness to the surrounding context. Furthermore, they use machine learning to combine these features to adjust the balance from document to document. Although Wikipedia Miner is not the first to use relatedness for Wikification [21], a critical lesson from its results is the importance of relatedness as a powerful feature for entity-linking tasks. These two concepts of commonness and relatedness are then refined by other approaches [84]. Our algorithm considers this feature by assigning weights to entity senses that indicate their similarity to their context. We run our experiments on the same datasets as Wikipedia Miner.

TagMe [45] proposes a voting scheme where the candidates for each mention can vote for all candidate entities of other mentions based on relatedness; candidates with higher prior probabilities have stronger votes. In contrast, in our proposed solution, the voting power of each candidate sense depends on its rank based on a previous voting round. Then TagMe uses two algorithms to decide the chosen sense for each mention: disambiguation by classifier (DC), which uses a probabilistic approach based on prior probability and relatedness to select the correct candidate, and disambiguation by threshold (DT), which makes a shortlist of the top candidates with relatedness above a predefined threshold, and then chooses the candidate with the highest prior probability among them. The WAT algorithm [154] is a more efficient version of TagMe [91], which is more sufficient in terms of complexity in comparison with TagMe. TagMe works well on short texts like Twitter posts, while our approach works well for short and longer texts.

GLOW [162] uses two sub-systems; a ranker and a linker. The goal of the ranker is to select the best candidate, and the linker decides if the recommended sense by the ranker is

good, or instead, switching the top-ranked disambiguation to null improves the objective function. The ranker sub-system uses two sets of features, local and global. Local features calculate the similarities between mentions and their candidate entities, incorporating the term frequency-inverse document frequency (TF-IDF) vectors for Wikipedia pages and the input text [73]. The global features measure the coherence among all candidate senses in terms of the sum of pairwise normalized Google distance (NGD) [32] and point-wise mutual information (PMI) [16] across all mentions in the whole disambiguation text. The linker sub-system is trained as a linear support vector machine (LSVM) to separate correct and incorrect linker outputs based on data collected from Wikipedia, which provides positive and negative examples for each mention according to Wikipedia's gold standard.

In their experiments, the authors of GLOW show that TagMe cannot process the whole document text at once, meaning that they attempted to input test documents one sentence at a time to TagMe since TagMe is designed for short text Wikification. Disambiguating each sentence independently resulted in poor performance using TagMe, where GLOW outperforms TagMe.

In the Wikisim project [173], the idea was splitting the text into disjoint windows for more accurate wikifying longer text, and the candidate senses for each mention in a window were ranked based on their average relatedness to all candidates of other mentions in the same window. Then, the confidence of each mention is calculated as the ratio of the scores of its top two candidates. The candidate with the highest confidence is then chosen as the key-entity for that window. All mentions in each window are then disambiguated based on the relatedness of their candidates to the key-entity. This approach improves the accuracy compared to TagMe while still missing information from other entities in neighboring windows. To address this problem, we consider context information around each mention using overlapping windows to achieve a higher coherence in the whole text than other approaches.

The state-of-the-art Wikifier is RedW [191] and is based on mapping the *longest sub-string match* between the mention and the Wikipedia entity titles. RedW creates a table of all Wikipedia titles. RedW tries to match the ambiguous mentions of n-grams to the table for every text. Unlike TagMe and Glow, RedW does not consider global features such as coherence. However, TagMe is designed for short text Wikification and is inferior to RedW compared to long texts (in terms of run-time), and Glow has not been compared against this

$$\left.\begin{array}{r} T = [t_1, t_2, \cdots, t_t] \\ M = [m_1, m_2, \cdots, m_m] \end{array}\right\} \rightarrow C = [C_1, \cdots, C_m]$$

$$= \begin{bmatrix} c_1^1 & \mathbf{c}_2^1 & \cdots & \mathbf{c}_m^1 \\ \mathbf{c}_2^1 & \mathbf{c}_2^2 & \cdots & \mathbf{c}_m^2 \\ & & \vdots & \\ \mathbf{c}_1^k & \mathbf{c}_2^k & \cdots & \mathbf{c}_m^k \end{bmatrix} \rightarrow E = [e_1, \cdots, e_m]$$

Figure 3.1: Demonstration of the Wikification problem. Text T is our input text with a list M of ambiguous mentions. The task is finding E, including the correct chosen entity e_i for each mention.

scheme. However, experiments show our proposed solution outperforms RedW, considering the text's coherence.

3.2.1 Formal Definition of Wikification Problem

We follow the same notations as in Wikisim [173]. We represent a text as a list of terms $T = \{t_1, ..., t_L\}$. We also consider a key-value inventory $I = \{(\ell_i, [e_1, \cdots, e_{k_i}])\}$, where each lemma key ℓ_i is a sequence of terms that can be matched to a Wikipedia page title, and values are lists of potential associated Wikipedia entities. A "mention" is then defined as a lemma key from this inventory in the text. A mention (phrase) list $M = \{m_1, ..., m_n\}$ is associated with every text, where $m_i = \{t_{r_i}, ..., t_{s_i}\}$ and $s_i < r_j$ for every $i < j$, so mentions do not overlap. Each mention m_i is either one term t_i from T, or a combination of consecutive terms from T, and each m_i can have k_i potential senses (or candidate entities). The set of these candidate senses is denoted as C_i. The goal is to find set $E = \{e_1, ..., e_n\}$, where e_i is the correct sense of mention m_i represented as a link to the corresponding Wikipedia page, see Fig. 3.1.

3.3 Methodology

As mentioned in the introduction, one of the steps in the Wikification process is mention detection, and another is disambiguation.

Mention Detection Step

Mention detection takes as input the text and a dictionary of known terms and finds

occurrences of these terms in the text [100]. In this work, we used the output of a recent Wikifier system (Wikisim) for this mention detection step [173]. Wikisim uses a finite state transducer to extract mentions using a long-dominant right heuristic. In the long-dominant right heuristic, the longest string is chosen right before mentions overlap [5], and hence avoids detecting overlapping mentions. Later, this model takes a dictionary of Wikipedia page titles and matches it to the input text. Most of the definitions and formulas notations present in this chapter are based on our baseline, the Wikisim project [173][6].

Disambiguation Step

The disambiguation step aims to find the correct sense for each ambiguous mention. We start with our Wikisim baseline approach and follow their definition, rules, and notations. Then, we modify Wikisim to improve its accuracy. As indicated by Sajadi et al., each mentions m_i can have k_i potential senses (or candidate entities) [173]. The set of these candidate senses is denoted as C_i. The goal is to find the set $E = \{e_1, ..., e_n\}$, where e_i is the correct sense of mention m_i, represented as a link to the corresponding Wikipedia page, see Fig. 3.1. Our contribution is in this step. For candidates, the list of the k most frequently linked entities (in the Wikipedia knowledge base) from each mention is selected to choose the best match between them. This is called a *popularity feature*. This feature has been defined as "the frequency of a given entity being linked to by the given mention" by Sajadi et al. [173]. In our work, $k_i = k, i = 1, \ldots, m$; meaning for every mention, we extract the same number of senses from Wikipedia. Our experiments show that when $k = 10$, the list includes the correct entity for each mention for about 85% of the mentions. The range we consider in experiments is $k \in [5, 15]$, and this choice is based on a similar observation in the Wikisim baseline approach. If the number of candidates for a mention is less than k, then the algorithm considers k equal to the number of all possible candidates for that mention.

When disambiguating a mention, we can consider different features; for instance, *popularity*, *context coherence*, and *key-entity coherence*. Most of the previous works considered the popularity feature [162, 173], but in this case, the less-popular candidates for entities are not covered. In our work, we propose a weighting schema that assigns weights to all the senses of a mention, so in this manner, even less popular senses have the opportunity to be compared with the context. If the similarity of these less popular senses of a mention with the context is higher than other popular senses, they will get chosen as the correct sense.

[5] https://github.com/OpenSextant/SolrTextTagger

[6] Most of the definitions present in this chapter are from our baseline, Wikisim project [173].

3.3.1 Notation and Definitions

We use the following notation and definitions, following Wikisim baseline [173]. Given a text document as input with a specified set of ambiguous mentions in it, CACW wikifies the ambiguous mentions based on these four key features:

1. Key-Entity Recognition
2. Overlapping Windows
3. Sense Ranking based on Assigned Weights
4. Text Coherence Maximization

The first feature was introduced in Wikisim [173]. Our first contribution is considering co-occurring mentions in consecutive windows, enabling us to augment the contextual information to find the correct sense. This algorithm is presented as OWCW in Section 3.3.3. We then improved this algorithm further in Section 3.3.4, introducing the CACW, which works based on ranking (different than previous approaches) the senses of ambiguous entities in OWCW.

Text Coherence

Our entity linker relies on the concept of coherence between word senses. The goal is to choose senses for all mentions in a text that maximize the selected coherence measure.

The most popular way to measure the coherence of a text is defined by the sum of all pairwise "semantic relatedness" scores of the senses chosen for all mentions [173]. We use $r(s_1, s_2)$ to denote the relatedness between two senses s_1 and s_2. In this approach, the semantic relatedness for a given pair of mentions is calculated as the relatedness function $r(.,.)$ of the entities in a text [145]. A solution is a set $E = e_1, ..., e_n$ i.e, each m_i is resolved to e_i. The aim here is to maximize the coherence, which is shown in Wikisim [173] as:

$$E^* = \arg\max_{E \in (C_1 \times ... \times C_n)} \Sigma_{i<j} r(e_i, e_j) \tag{3.1}$$

Finding E^* by searching the whole space $(C_1 \times ... \times C_n)$ is computationally expensive. Therefore, different approaches were proposed to reduce the number of candidate solutions to be considered [174]. One of these approaches is based on Vector Space Modeling (VSM) of the *Lesk Like* method [173]. This method considers the context in which a mention m

occurs and selects the candidates based on their similarities with the context (the input text). To explain this method, we first need to review some definitions from Wikisim [173].

Definition 3.3.1. For any candidate c of a given mention, $R(c)$ is the candidate representation as a vector embedding of the corresponding Wikipedia entity. For the rest of this work, we use the Word2Vec representation[7]. Our proposed Wikifier is independent of the choice of embedding. In the experiment section, the same embedding was chosen for all methods, to have a fair evaluation.

Following our baseline Wikisim [173]:

Definition 3.3.2. The representation of m, $R(m)$, is the average of its candidate representations [173]

$$R(m_i) = \frac{1}{|C_i|} \sum_{c \in C_i} R(c) \tag{3.2}$$

Definition 3.3.3. Context vector w.r.t mention m, $\hat{R}(m)$, is the average of all other mention representations in the text (every mention exclude m):

$$\hat{R}(m) = \frac{1}{n-1} \sum_{m_i \in M \setminus m} R(m_i) \tag{3.3}$$

The context-vector disambiguation used in Wikisim [173] consists of:

- "Calculate mention representations $R(m_i)$ for all $i = 1, ..., n$",
- "Extract context vectors $\hat{R}(m_i)$ for all $i = 1, ..., n$"
- "The disambiguated entities e^* are those with maximum cosine similarity to their context":

$$E^* = [\text{arg max}_{c \in C_i} R(c).\hat{R}(m_i) | i = 1, ..., n] \tag{3.4}$$

There is a conjecture introduced by Lazi et al. that states there is a single context word that suffices to disambiguate all mentions in a text [91]. By using this conjecture, it has

[7] Any other appropriate embedding could be used to replace the Word2Vec embedding.

been demonstrated that, for a given text, there exists one entity among candidates of one of the mentions in the text, referred to as the *key-entity* and denoted by e^*, that can help to disambiguate the rest of the mentions [173]. The senses of all mentions are chosen to maximize the total pairwise relatedness between the key-entity and the mentions.

Key-Entity Recognition

Each mention m_j has a different number of possible candidate senses from 1 to k_j, indicated by the set C_j in Algorithm 1. By comparing the representation of each candidate's sense of a mention to the representation of the mention's context, we can have a ranking of these candidate senses according to their relevance within this context. Now assume that for each mention m_i, k_i^1, and k_i^2 are the best and the second-best candidates according to this ranking. Considering the inner product as the similarity measure, which can effectively capture co-occurrence information versus other measures, here are the calculations for k_1 and k_2:

$$k_i^1 = \arg\max_{t \in C_i} R(t).\hat{R}(m_i), \tag{3.5}$$

$$k_i^2 = \arg\max_{t \in C_i \setminus \{k_i^1\}} R(t).\hat{R}(m_i) \tag{3.6}$$

Definition 3.3.4. The confidence value of m_i is the proportional difference between the relatedness score of k_i^1 and k_i^2 to the context, based on [173]:

$$\text{conf}(m_i) = \frac{R(k_i^1) \cdot \hat{R}(m_i) - R(k_i^2) \cdot \hat{R}(m_i)}{R(k_i^2) \cdot \hat{R}(m_i)}$$

The key-entity e^* is the top candidate k_i^1 of the mention m_i with the highest confidence:

$$e^* = k_i^1 \text{ where } i = \arg\max_{i \leq n} \text{conf}(m_i) \tag{3.7}$$

The intuition of why this is a good way to choose e^* is directly related to the confidence score. The maximum confidence means the mention has the highest margin between its first and second best senses (related to the input text as context).

The entities with the true meaning, based on Wikisim, are those whose cosine similarities

to $R(e^*)$ are maximum.

$$E^* = \arg\max_{t \in C_i} R(t) \cdot R(e^*), \quad i = 1, \cdots, n \tag{3.8}$$

This method is much faster; chunking the text into smaller windows significantly reduces the number of word pairs that need to be compared. In the naive approach, comparing every pair of words results in a quadratic increase in computational complexity as the text length grows. This approach also achieves better accuracy (5% better) than the naive approach based on the pairwise relatedness between all pairs of words [173]. A common approach for long texts is to divide the text into fixed-sized windows and independently disambiguate the mentions in each window. The assumption is that if the windows are large enough, each window contains a meaningful key-entity [91]. The motivation is the choice of a more localized key-entity to improve the accuracy. The algorithm benefits from a contextual focus within each window. Considering a smaller window size as context can better capture the specific meanings of words within that context. In contrast, the naive approach considers the entire text, which may introduce ambiguity and unnecessary complexity when words have multiple meanings in different parts of the text.

3.3.2 Coherence Measure

Definition 3.3.5. In this study, we defined **coherence measure** to measure how coherent the text is after disambiguation. As mentioned, each entity has a representation, $R(e)$. When the disambiguation procedure is done, the algorithm has selected one candidate sense –with representation $R(c_i)$–for each mention m_i. After the disambiguation, we can measure how coherent the text is using selected candidate senses for each mention. We define the text's coherence by the average cosine pairwise similarity between the selected senses of all mentions. To compute the average, we divide the sum of similarities between all pairs by the number of pairs, $\binom{m}{2}$. This new coherence metric is based on the idea of sense representation.

$$Coherence(T) = \frac{\Sigma_{i,j=1,i\neq j}^{m} cos(R(c_i), R(c_j))}{\binom{m}{2}} \tag{3.9}$$

We use this measure to compare the coherence of the disambiguated text with our method and the baseline methods. **Disambiguated Text** is the annotated text, including Wikipedia

pages of the senses chosen for all ambiguous mentions [174].

3.3.3 OWCW: Overlapping-Windows Concept Wikifier Algorithm

The Wikisim approach [173] divides the text into disjoint windows and performs disambiguation by key-entity selection independently in each window. The novel idea of our Wikifier came from the defect of this last coherence-based approach. As we verify experimentally, this separate window-based approach may incorrectly disambiguate mentions if their "optimal" key-entity is in a neighboring window; thus, it is still close to the mention but "invisible" due to the disjoint partitioning. Their method segments the input text and uses these disjoint text windows as the context to disambiguate each entity. Similarly, our algorithm refers to the windows as the context for each entity. By considering the windows as a sentence, a paragraph, or a specific text length, i.e., m mentions, our algorithm performs best when $m = 20$ in the datasets that we used in our experiments. The defect with the previous approach is ignoring the information of the neighboring contexts when disambiguating entities. To clarify this problem, we consider the following running example.

Example: How Neighboring Contextual Information is important in Disambiguation

Considering this sentence, "Mars, galaxy, and bounty are all chocolate," with mention list M=[mars, galaxy, bounty, chocolate]. By segmenting the text into disjoint windows of size 3, as shown in Fig. 3.2(A), the windows are:

W_1= [Mars, galaxy, bounty],

W_2= [chocolate].

Wikisim approach disambiguates "mars" as "Mars-planet", "galaxy" as "galaxy- astronomical -system", "bounty" as "bounty-bay", and "chocolate" as "chocolate-foodstuff". The first three mentions are assigned incorrect senses. In the above example, the correct key-entity "chocolate-foodstuff" which is necessary to choose the correct senses, is in a different window. With CACW, we use overlapping windows to address this problem while imposing only a modest increase in the running time.

Our main contribution is managing the Wikification problem by looking for more information around each mention. We achieved this goal by implementing sliding windows across the long text, finding correct senses based on each window's key entities, and maximizing similarities between chosen senses for each mention of the same window.

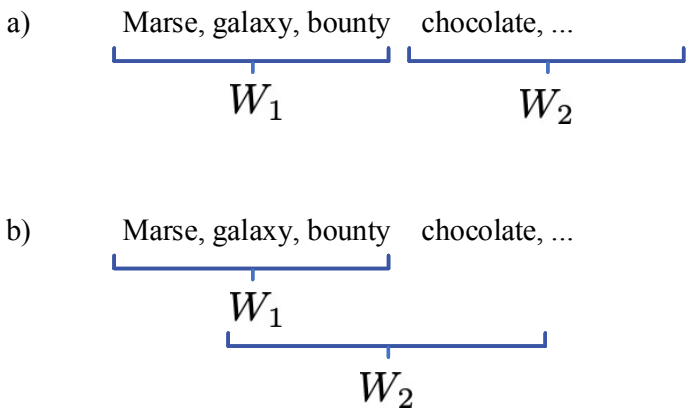


Figure 3.2: Partitioning the extracted mentions of sentence "Mars, galaxy, and bounty are all chocolate" into separate windows of size 3 in (a), and partitioning it into overlapping windows of size 3 with offset 1 in (b).

In our method, we calculate semantic relatedness by $r(.,.)$ function, line 8 in Algorithm 1, which is the cosine similarity of two representation vectors. We use cosine similarity due to its better performance in comparison with other similarity metrics [26]. We partition the text document (**D**) into smaller segments by defining a window-size variable (**WS**) and overlap two consecutive windows by an offset variable (**OFF**) as inputs of the Algorithm 1. This segmentation idea is shown in Fig. 3.3.

We assign the key-entity in each window, line 4 of Algorithm 1, as defined in Eq. 3.8 and disambiguate each mention with a sense that has maximum similarity with that key-entity, line 9-10 in Algorithm 1.

As mentioned, segmenting the text into fixed-size windows is controlled by changing the variable WS. Once we have the key entities of each window, we use another variable offset. The offset indicates the size of the mention list mines the number of common mentions between two consecutive windows. For example, if window-size is 20 and offset is 19, the first window covers m_1 to m_{20}, and second window covers m_{20} to m_{39}, so these two windows overlap only one mention which is m_{20}. Using offset, sliding each window over the next neighboring window by using offset, allows one to consider more information around each mention to disambiguate it. Considering these overlapping windows makes the accuracy of our method better since it results in the appearance of mentions in more than one window, except for the first and the last mentions. Due to the overlap, those mentions belonging to multiple windows will be compared with each key-entity in their windows,

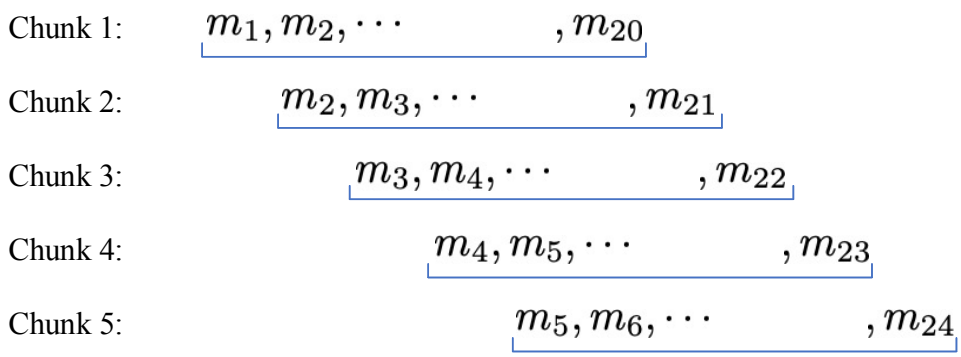


Figure 3.3: Segmenting the text into overlapping windows. In this figure, our text has 24 mentions. The considered window's size is 20, and the Offset size is 1; it means two consecutive windows overlap 19 common mentions, and we have 5 windows in total.

Algorithm 1: OWCW. The input text document D is chunked into smaller partitions as windows using the window size variable WS. Overlapping two consecutive windows by offset variable OFF. OWCW chooses the key-entity in line 4 by Eq. 3.7 and disambiguates each sense by maximum similarity with the key-entity of the same window, as mentioned in lines 6 to 9.

Input: $D, M, WS, OFF, C_1, \cdots, C_n$
Output: E

1 $W \leftarrow$ Set of windows obtained by segmenting D based on WS and OFF
2 $\hat{E} \leftarrow \emptyset$
3 **for** $W_i \in W$ **do**
4 $e_i^* \leftarrow$ key entity of W_i
5 **for** $m_j \in M$ **do**
6 **for** $W_i \in W,\ m_j \in W_i$ **do**
7 **for** $c_{jl} \in C_j$ **do**
8 $s_{(l,i)} \leftarrow r(R(c_{jl}), R(e_i^*))$
9 $\hat{e}_{ij} \leftarrow e_\ell$ where $\ell =$ arg max$s_{(l,i)}$
10 $\hat{E}$.append arg$(max_i(\hat{e}_{ij}))$
11 **return** $\hat{E}$

and the entity with the minimum distance to the key-entity will be considered as the correct meaning for that ambiguous word. For example, if we set the window size to 20 and the offset to 19 (choosing these variables is based on our experiments), then the input text's first and last mentions appear in one window. The second mention from the beginning of the text and the second mention from the end of the text appears in two separate windows. The same procedure applies to the appearance number of all the mentions in the windows. We consider one mention in one window and find the similarity of its candidate entities with the key-entity of the same window based on the maximum coherence measure method. We repeat the same procedure for finding the similarity of the same mention's candidate entities with the key-entity of other windows to which this mention appears. Finally, the candidate entity with the highest similarity across all relevant key entities is selected for this mention. Since we consider the similarities with more than one key-entity, this chosen entity is more likely to be coherent with the context of the entire input text.

Consider the example from the previous Section 3.3.3 and partition the text with a window size equal to 3 and an offset equal to 1, as shown in Fig. 3.2 (B). The windows are:

W_1= [Mars, galaxy, bounty],

W_2= [galaxy, bounty, chocolate].

In the disambiguation process, because we slide the windows, the sense of "chocolate-foodstuff" impacts the possible senses for mentions occurring in the window of "chocolate", i.e., mentions "bounty" and "galaxy". The similarity between "chocolate-foodstuff" and "bounty-chocolate-bar" is higher than the similarity of "chocolate-foodstuff" with "bounty-bay", so the disambiguated entity for "bounty" is "bounty-chocolate-bar" instead of "bounty-bay". This disambiguation is because of considering the comparison of "bounty" with "chocolate" that are appearing in the same window. This small example shows the power of overlapping windows, considering more information around each mention for disambiguation. The mention of "galaxy" is in the same window as "chocolate" and "bounty", so the algorithm considers its similarity with these two mentions and finds its most relevant sense as "galaxy-chocolate-bar" instead of "galaxy-astronomical-system". In the first window, we are still comparing the similarity of senses between "Mars", "galaxy", and "bounty". The cosine

similarity of the pairs (mars-astronomical-system, galaxy-astronomical-system) and (mars-astronomical-system,bounty-bay) are higher compared to (mars-astronomical-system,galaxy-chocolate-bar) and (mars-astronomical-system,bounty-chocolate-bar). So, we will mis-disambiguate the mention of "mars" in this sentence. By assigning higher importance weight to the second sense of "galaxy" and the second sense of "bounty", the algorithm has more potential to choose the correct sense of "mars". In the next section, we introduce a context-aware ranking of word senses that propagates information between windows and helps address the problem we just observed in the example.

3.3.4 CACW: Context-Aware Concept Wikifier

The second novel idea of our Wikifier is related to the defect of overlapping windows, as mentioned above. This problem happens when the difference between the top two senses ${k_i}^1$ and ${k_i}^2$ for each mention m_i is too high or too low, i.e., when two senses for the same mention are too close together or, on the other hand, are too far from each other. In such cases, the algorithm mis-disambiguates the correct sense for those mentions. In this section, we propose our second idea to address this problem.

The main difference between OWCW and CACW is in calculating the key-entity of each window. The key-entity for a window W is defined as the candidate entity with the highest confidence among all mentions in that window. In OWCW, the confidence is calculated based on the aggregate similarity of each entity with all candidate entities for any other mention in the same window. However, one should notice that not all candidate entities for each mention are equally relevant to the context. Therefore, assessing the significance of each candidate entity when selecting the key-entity by its relatedness to the context becomes advantageous. In light of this, we introduce Algorithm 2, which facilitates the allocation of weights to candidate entities for all mentions within a specific range by means of an iterative procedure

First, we initialize weights for entity candidates of a mention, line 3 of Algorithm 2. The aggregate weight of all candidate entities for each mention in the window is always 1. Then, from the second iteration, we update the weights by considering the weighted relatedness measure between each entity of a mention and all candidate entities for any other mention in the same window through lines 5 to 13 of Algorithm 2. We can think of it as applying the PageRank algorithm in a graph, where the vertices correspond to candidates $c_{i,j}$, and the

edges between them correspond to semantic relatedness values $r(\cdot, \cdot)$. In Algorithm 2, we show this weight assigning in the procedure of choosing the key-entity by taking care of the confidence value. Moreover, we applied the same weight-assigning procedure for ranking sense entities of all other mentions. The T in Algorithm 2 denotes the number of iterations through which we update all the weights for all entities in window W. I_W denotes the set of all indices of mentions $m_i \in W$. Finally, we use e^*_W to denote the selected key-entity for window W.

By assigning these weights to the senses of our example in Section 3.3.3, the second sense for "galaxy" and "bounty" have higher ranks because of similarities from the second window. As a result, when finding similarities in the first window, "mars-chocolate-bar" is closer to "galaxy-chocolate-bar" instead of "mars-planet", so this sense of "mars-chocolate-bar" has maximum similarity. By applying this algorithm to this sentence, the disambiguated senses for mentions include "mars" as "mars-chocolate-bar", "galaxy" as "galaxy-chocolate-bar", "bounty" as "bounty-chocolate-bar", and "chocolate" as "chocolate-foodstuff". So, all the mentions are disambiguated correctly. This example shows how the proposed CACW algorithm corrects disambiguation errors due to the separate windows in the past coherence-based approaches and finds the best meanings for each mention based on the context of the input document.

3.4 Evaluation

We now explain the experimental setup of our evaluation on the disambiguation task for our proposed Wikifier. This section provides details of the datasets we used, the configuration for our model and variable setups (including different window sizes and offsets), characteristics of our system, baseline approaches that we compared with, and lastly, we talk about measurement metrics for the evaluation. In this work, we follow the usual convention for defining metrics for disambiguation [7]. In particular, *accuracy* is the proportion of mentions that were disambiguated correctly. The specification of the machine that is used for running the experiments is as follows: Intel(R), Xeon(R), CPU E5-2650, 2.00GHz, 64 bits, with 256 GB RAM and 8 cores.

Algorithm 2: CACW: Iterative Weighted Key-Entity Selection. Assigning weights to the candidate entities for each mention in a window. It starts with initial weights (lines 1-3), then goes through an iterative process to assign those weights based on similarities between two entities (in the while loop).

Input: $\{m_i,\ i \in I_W\},\ C_i,\ T$
Output: e^*_W

1 **for** $i \in I_W$ **do**
2 **for** $c_{i,j} \in C_i$ **do**
3 $w_{i,j} = 1/|C_i|$
4 $t = 0$
5 **while** $t < T$ **do**
6 t += 1
7 **for** $m_i \in W$ **do**
8 **for** $c_{i,\ell} \in C_i$ **do**
9 temp = 0
10 **for** $m_j \in W$ **do**
11 **for** $c_{j,k} \in C_j$ **do**
12 temp += $\frac{r(c_{i,l},c_{j,k})}{\sum_{i'}\sum_{l'} r(c_{i',l'},c_{j,k})} w_{j,k}$
13 $w_{i,\ell}$ = temp
14 **for** $m_i \in W$ **do**
15 $a_i = \arg\max_{j\in\{1,\cdots,|C_i|\}} w_{i,j}$
16 $b_i = \arg\max_{j\in\{1,\cdots,|C_i|,\ j\neq a_i\}} w_{i,j}$
17 conf(i) = $\left(w_{i,a_i} - w_{i,b_i}\right)/w_{i,b_i}$
18 $\ell = \arg\max_{i\in I_W}$conf$(i)$
19 **return** $C_\ell[a_\ell]$

3.4.1 Evaluation Datasets

We evaluate our algorithms on the datasets for the disambiguation task with Wikipedia, which is the standard dataset for the task as stated in the Wikification literature [7]. Each dataset is organized in the same way, just with different types of content. Each record contains two fields, text, and mentions. The text field comes pre-tokenized and contains all of the text as a list. The mentions field is a list of pairs containing the index of the mention in the text field and the Wikipedia page ID to link to [173].

The datasets consist of two English news collections: AQUAINT and MSNBC. Additionally, there is another dataset called Kore, which contains sentences curated by humans. AQUAINT and MSNBC are texts from the news and are generally more straightforward to disambiguate due to the formality of the text. Finally, we have a dataset based on Wikipedia that uses the opening paragraphs of various Wikipedia pages. We performed an evaluation on Wiki5000 and Wiki30000, which consist of 5000 and 30000 articles on Wikipedia (based on the order of article creation). These datasets were used to evaluate previous wikifiers [154, 173].

AQUAINT is a corpus of 50 documents taken from three news sources: the Xinhua News Service (People's Republic of China), the New York Times News Service, and the Associated Press World Stream News Service. This corpus has been used in official benchmark evaluations conducted by the National Institute of Standards and Technology (NIST). The data files contain roughly 50 texts, 10667 words, and 727 mentions. The sampling for this corpus covers the period from January 1996 to September 2000, inclusive, for the Xinhua text collection, and from June 1998 to September 2000, inclusive, for the New York Times and Associated Press [125].

MSNBC is a corpus of 10 selected news categories of Business, U.S. Politics, Entertainment, Health, Sports, Tech Science, Travel, TV News, U.S. News, and World News. From each category, 2 top stories are annotated by a human. The data consists of 20 news document texts, 10903 words, and containing 656 mentions in total [35].

Kore consists of 50 human-curated hard-to-disambiguate sentence texts that contain relatively short texts with few mentions. The number of words is 769, and the number of mentions is 192. This dataset is handcrafted to make it more complicated. Most of the mentions in this dataset are less popular mentions. On the other hand, in the news datasets, most of the mentions are those with popular correct senses [63, 173].

Wiki5000 is the same dataset used in the Wikisim project. This dataset includes 265974 words, and the number of mentions is 30612. Wiki5000 is the first 5000 articles of Wikipedia, based on the order of articles created, ordered by their IDs. The opening paragraph of each article has been chosen to create this dataset since entities are not necessarily linked throughout the whole article [173].

Wiki30000 includes articles from Wikipedia that do not overlap with Wiki5000 [230]. The number of texts is 30000, the number of words is 1431271, and the number of mentions is 169880. Wiki30000 has been produced in the same way, similar to Wiki5000. This dataset includes the next 30000 articles on Wikipedia after Wiki5000. The article IDs used in this data include article IDs from 5001 to article ID number 35000.

All these mentioned datasets are non-specific subject datasets. At the same time, one could ask how likely our algorithm is correct on a subject-specific dataset, for example, in the biomedical domain. To test this, we run experiments on the MSH WSD dataset[8]. MSH WSD is the result of developing a WSD test collection using the Unified Medical Language System (UMLS) Metathesaurus [71]. MSH WSD consists of 106 ambiguous abbreviations, 88 ambiguous mentions, and nine of both, for a total of 203 ambiguous words. Each instance containing the ambiguous word is assigned a CUI (Concept Unique Identifier) from the 2009AB version of the UMLS [12]. For each ambiguous mention/abbreviation, the data set has a maximum of 100 instances per sense obtained from MEDLINE[9], totaling 37,888 ambiguity cases in 37,090 MEDLINE citations [195].

3.4.2 CACW Configuration and Parameters of Test Evaluation

We experiment with window sizes (WS) between 5 and 40 with a step of size 5 in our algorithm. Also, we change the offset between 1 to $|\text{WS}| - 1$. The performance of our proposed Wikifier for different window sizes is shown in Fig. 3.4 on the Wiki30000 dataset, which is our largest dataset. This figure reports three different offset values, including 1, $|WS|/2$, and $|\text{WS}| - 1$. The result shows that $\text{WS} = 20$ is the best choice in terms of accuracy for all the offset values. We also set the number of iterations in Algorithm 2 to be 10 (e.g., T = 10).

[8] https://lhncbc.nlm.nih.gov/ii/areas/WSD/collaboration.html

[9] Medical Literature Analysis and Retrieval System Online (MEDLINE) is a bibliographic database that provides access to a vast collection of biomedical literature.

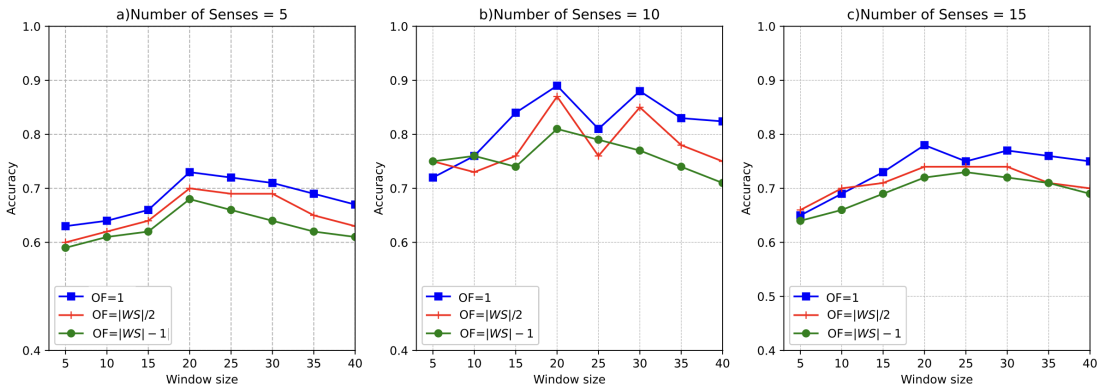


Figure 3.4: The effect of Window Size |WS| on CACW's accuracy, using "Wiki30000" dataset, with different offset (OF). In a) the number of senses for each mention is set to 5. In b), the number of senses for each mention is 10, and in c), the number of senses for each mention is set to 15.

3.4.3 Baseline Wikifiers

We compare our Wikifier with two knowledge-based entity linkers: **Wikisim** [173], which is the most recent key-entity Wikifier, and **TagME** [45] which is available as a web service [10]. We also compare with **GLOW** [162] and **Wikipedia Miner** [125] since we have their accuracies on three common datasets. Lastly, we compare our approach's accuracy and run time with RedW [191], which is a context-free Wikifier. In the experiments, the initial list of ambiguous candidate mentions is the same for the evaluated baselines to preserve the fairness of comparison.

3.4.4 Evaluation Measure

The proposed algorithms and approaches for Wikification, as mentioned above, were evaluated by using standard information retrieval (IR) measures, namely **precision**, **recall**, and the **F1-measure** [7]. In particular, we evaluated the Wikifier as a multi-class classifier, where classes are the possible senses (i.e., entities in Wikipedia). Each mention is to be classified by the Wikifier in its correct sense. To consider the usual IR metrics, we need to define true positive, true negative, false positive, and false negative concerning each class (i.e., sense). We follow the same definitions Sebastiani provided in [185] as follows. This result is reported in the following.

Considering a particular sense, namely sense i, we compare the result of Wikification of

[10] https://tagme.d4science.org/tagme/

Candidate Sense c_i		Expert Decision	
		Yes	No
Algorithm Decision	Yes	TP_i	FP_i
	No	FN_i	TN_i

Table 3.1: The definition of True/False, Positive/Negative concerning a given sense, i-th candidate sense (c_i), from Wikipedia. The source of this table is [185].

one ambiguous mention by the algorithm and the expert, as shown in Table 3.1, with the following definitions of each notation.

- True Positive: A mention that is correctly disambiguated to sense i by the algorithm, while its correct sense based on the expert's opinion is sense i.
- False Positive: A mention that is disambiguated to sense i by algorithm, while based on the expert's opinion, its correct sense is any other sense different from sense i.
- False Negative: The correct sense of the mention based on the expert's opinion is sense i, while the algorithm disambiguates this mention to a different sense.
- True Negative: The correct sense for the ambiguous mention is not the i sense, and the algorithm does not pick sense i either.

For precision, recall, and F1 measures, we used definitions as mentioned in [108, 84, 87], which are as follows.

- **Precision (P)** is the ratio of correctly disambiguated mentions to all disambiguated mentions.
- **Recall (R)** is the ratio of correctly disambiguated mentions to the total number of mentions to be disambiguated.

  Note that the total mentions being disambiguated can differ from the total number of disambiguated mentions by the Wikifier since the mention detection might fail to detect some of the ambiguous mentions.
- **F-measure** is the harmonic mean of precision and recall.

$$F = \frac{2 \times P \times R}{P + R} \tag{3.10}$$

In addition to the recall, precision, and F1, we used micro-averaged precision and macro-averaged precision as below:

$$\hat{\pi}^{\mu} = \frac{\Sigma_{i=1}^{s} TP_i}{\Sigma_{i=1}^{s} (TP_i + FP_i)} \tag{3.11}$$

$$\hat{\pi}^{M} = \frac{\Sigma_{i=1}^{s} \hat{\pi}_i}{s} \tag{3.12}$$

where TP_i denotes the number of true positives with respect to sense i, FP_i the number of False Positives with respect to sense i, $\hat{\pi}_i$ is the precision with respect to sense i, and s is the size of the collection of all senses for the mentions existing in our document.

3.5 Results

Our experiments aim to explore the effectiveness of the proposed approach for the Wikification problem. The hypothesis of engaging more contextual information from knowledge bases and the input text simultaneously to disambiguate entities in a way that is more related to the context after disambiguation is supported by the results in Table 3.2. In this experiment, we compared the baselines of GLOW, Wikipedia Miner, TAGME, Wikisim, RedW, and our two proposed approaches, OWCW, and CACW, on all the evaluation datasets mentioned in Section 3.4.1. On the Kore dataset, the accuracy of the GLOW is 69% which is close to the accuracy of Wikipedia Miner with an accuracy of 71%, but better than Wikisim, which is 63% accurate. One of the key features of Wikipedia Miner is its use of a large knowledge base of Wikipedia pages and its graph structure. Using this graph structure helps to disambiguate more accurately in comparison with GLOW, which uses NGD and PMI across all the ambiguous mentions. Between our two models, CACW has an accuracy of 78% and OWCW has 72% accuracy, which shows how the ranking improves the results of disambiguation. On the AQUAINT dataset, Wikisim is 64% accurate, and TAGME has an accuracy of 66%. GLOW and Wikipedia Miner have close accuracies of 72% and 74%, respectively. OWCW is also close to these two approaches with an accuracy of 73%, while CACW is 79% accurate. On this dataset, RedW has the highest accuracy of 83%, which shows how this model performs well on short documents. On the MSNBC dataset, TAGME has an accuracy of 61%, Wikisim has an accuracy of 65%, Wikipedia Miner is 68% accurate, and GLOW has an accuracy of 74%. The accuracy of our first model, OWCW is 77% and

CACW has an accuracy of 81%.

	Method	Kore	AQUAINT	MSNBC	Wiki5000	Wiki30000
P	GLOW	0.69	0.72	0.74	0.65	0.66
	Wikipedia Miner	0.71	0.74	0.68	0.68	0.69
	TAGME	0.74	0.66	0.61	0.57	0.58
	Wikisim	0.63	0.64	0.65	0.61	0.60
	RedW	-	**0.83**	-	0.57	-
	OWCW	0.72	0.73	0.77	0.79	0.88
	CACW	**0.78**	0.79	**0.81**	**0.85**	**0.93**
R	GLOW	0.68	0.74	0.73	0.67	0.66
	Wikipedia Miner	0.70	0.73	0.67	0.67	0.68
	TAGME	0.63	0.64	0.51	0.64	0.65
	Wikisim	0.67	0.62	0.53	0.55	0.56
	RedW	-	**0.79**	-	0.67	-
	OWCW	0.75	0.76	0.75	0.77	0.87
	CACW	**0.79**	**0.79**	**0.79**	**0.81**	**0.89**
F1	GLOW	0.68	0.72	0.73	0.65	0.66
	Wikipedia Miner	0.70	0.73	0.68	0.67	0.70
	TAGME	0.68	0.64	0.55	0.60	0.61
	Wikisim	0.64	0.62	0.58	0.57	0.57
	RedW	-	**0.81**	-	0.62	-
	OWCW	0.73	0.74	0.75	0.78	0.87
	CACW	**0.78**	0.78	**0.80**	**0.83**	**0.91**

Table 3.2: Precision, Recall, and F1-measure on the Wikification task of unsupervised knowledge-based approaches of GLOW, Wikipedia Miner, TAGME, Wikisim, and RedW in comparison with our proposed algorithms OWCW and CACW.

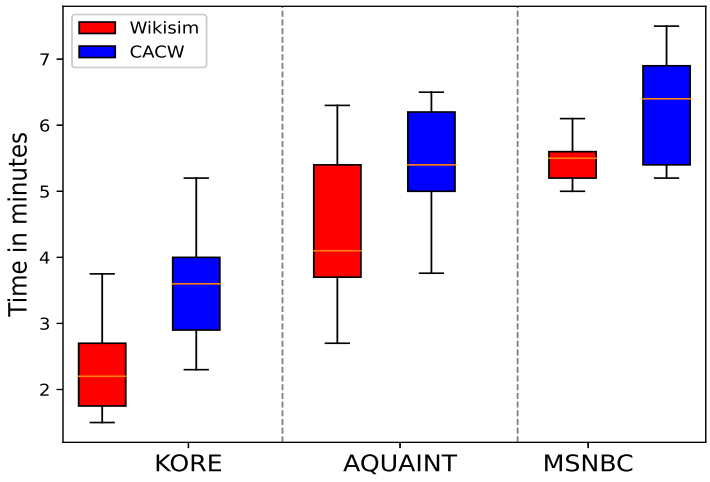
The performance of GLOW on Wiki5000 and Wiki30000 is similar, which is 65% and 66%, respectively. Similarly, the performance of Wikipedia Miner on these two datasets is similar, which is 68% on Wiki5000 and 69% on Wiki30000. The accuracy of TAGME and Wikisim are close to each other, near 58%. The RedW model has the lowest accuracy of 57%, which shows this model does not perform well on longer documents. The performance of OWCW on Wiki5000 is 79% and on Wiki30000 is 88%, while the performance of CACW is 85% and 93% on these two datasets. Using CACW, we are more confident that we are considering all relevant information in the document for disambiguating the ambiguous mentions. The improvement in our results over the baselines is statistically significant (χ^2 test with $p < 0.05$). The reported result shows that CACW surpasses all approaches on short and longer texts, except on the AQUAINT dataset, which RedW performs better. At the same time, on longer documents, the entity disambiguator of CACW with assigning weights

works better than the title-matching approach of RedW on a long corpus. The reason is using the title matching method of the RedW, which gives a shorter range of possible candidates for each ambiguous mention. To summarize, RedW works well on short documents with a limited number of ambiguous mentions, while CACW works better on long input text.

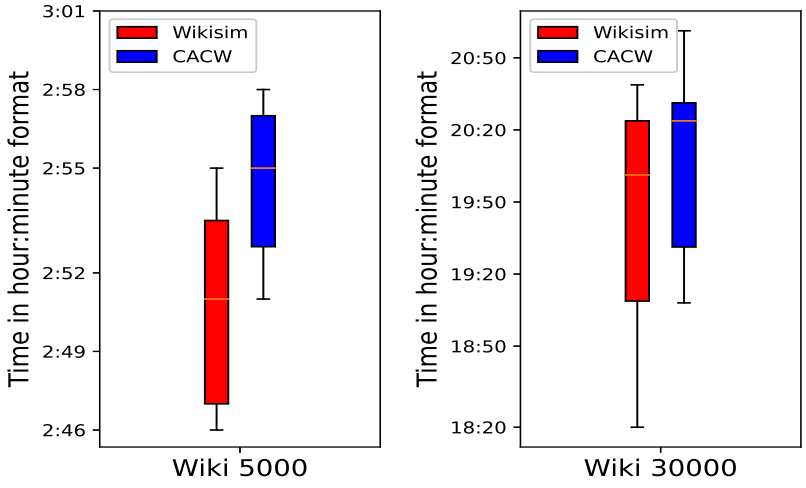
To compare the computation cost of CACW with the baselines, we run another experiment using disjoint windows and overlapping windows. The results are reported in Fig. 3.5. We show differences between run times of these settings in Fig. 3.5; The red boxes are for the disjoint window method, Wikisim, while the blue boxes are for our approach, CACW. We conclude that the complexity of CACW increases slightly more than the Wikisim algorithm. Also, the performance comparison of these two approaches is presented in Table 3.3. This table shows the accuracy and the run-time of the OWCW algorithm and the Wikisim baseline algorithm on three datasets, including Kore, Aquaint, and MSNBC. We vary the size of windows from 5 to 10 in this experiment, meaning that we have five mentions in each window in the first run, and in the second run, we have ten mentions in each window. The run-time for each experiment is reported as well in this table, which is shown as T, as the third row in each window size set-up. The results of this table show that OWCW's accuracy is better than Wikisim on the Kore, Aquaint, and MSNBC datasets. At the same time, we observe the run time of the OWCW has increased compared with the Wikisim run time, but it is not a dramatic increase in the cost of the method, making it a good approach to apply. In general, as we observe, when the window size is increased, the performance of the proposed model is better, while it still varies between datasets with different sizes.

The proposed algorithm leads to a new Wikifier with increased accuracy, at the cost of a modest increase in computational complexity, compared with RedW [191] and Wikisim. The comparison between the run-time of these schemes is shown in Table 3.4. The comparison of these models is implemented over two sets of datasets, the AQUAINT dataset as a short corpus and Wiki5000 as a long corpus. With the suggested method, more than 5% performance improvement is achieved consistently across datasets.

The accuracy of the CACW algorithm on the biomedical dataset is 68%, which is not as good as this algorithm's performance on the news dataset. One reason for this poor



(a)



(b)

Figure 3.5: Run-time comparison of the Wikisim approach and CACW approach, by changing window size; WS= 3, 5, 10, 15, 20, 25, 30, 35, 40, and offset=1, each run. In (a), we compare the models on the short texts, while (b) represents the results of comparing the models on the averaged length text, which is Wiki5000. The subfigure (b) on the right displays the results of comparing the models on the larger dataset, the Wiki30000. This result shows that while the new algorithm's accuracy increases, the run-time slightly increases compared to the Wikisim algorithm.

Window Size	Measurements	Approach	Kore	Aquaint	MSNBC
|WS|=5	$\hat{\pi}^{\mu}$	Wikisim	.443	.631	.642
		OWCW	.451	.653	.650
	$\hat{\pi}^{M}$	Wikisim	.435	.630	.687
		OWCW	.571	.651	.701
	T	Wikisim	1:32	6:03	6:01
		OWCW	2:17	19:21	38:33
|WS|=10	$\hat{\pi}^{\mu}$	Wikisim	.443	.641	.639
		OWCW	.572	.753	.762
	$\hat{\pi}^{M}$	Wikisim	.435	.639	.698
		OWCW	.592	.711	.732
	T	Wikisim	1:33	6:27	7:12
		OWCW	4:15	8:16	9:03
|WS|=15	$\hat{\pi}^{\mu}$	Wikisim	.471	.645	.643
		OWCW	.598	.798	.785
	$\hat{\pi}^{M}$	Wikisim	.445	.641	.699
		OWCW	.631	.789	.823
	T	Wikisim	1:41	6:35	7:56
		OWCW	5:45	12:22	14:35
|WS|=20	$\hat{\pi}^{\mu}$	Wikisim	.477	.668	.687
		OWCW	.688	.802	.821
	$\hat{\pi}^{M}$	Wikisim	.449	.648	.702
		OWCW	.689	.812	.867
	T	Wikisim	1:54	7:43	8:14
		OWCW	6:05	14:27	15:46

Table 3.3: Comparing disambiguation results of baseline Wikisim approach (non-overlapping window approach) with the proposed approach (overlapping windows approach): Micro Averaged Precision ($\hat{\pi}^{\mu}$) and Macro Averaged Precision ($\hat{\pi}^{M}$), across different candidate numbers and datasets. The precision of the proposed approach (OWCW) is higher than baseline (Wikisim) across different window sizes (WS) and different datasets. The last row of T shows the run-time.

Model	Short corpus	Long corpus
RedW	67 s	554 s
Wikisim	81 s	902 s
CACW	135 s	1412 s

Table 3.4: Run-time performance of RedW, Wikisim, and CACW on AQUAINT dataset as the short corpus, and Wiki5000 as the long corpus, reported in seconds (s).

performance on the biomedical dataset would be because of using the English Wikipedia general text as the knowledge base and not a domain-specific biomedical knowledge base, as shown in domain-specific works [148]. While we are using Wikipedia as the knowledge base, our method works well on general English text. A combination of our approach with biomedical text embeddings and a biomedical-specified knowledge base, like UMLS, would be one possible future direction to improve the results.

The other aim was to identify the best window size and the best offset in each dataset based on the text's size and subject. After testing this idea on different datasets, estimating the relationship between window size and offset was impossible since disambiguation depends on the context, which varies in every subject. In the algorithm, similar to the Wikisim setting for a fair comparison, we considered window sizes of 5, 10, 15, 20, 25, 30, 35, and 40 and limited the number of possible senses for each mention to 5, 10, and 15. Also, we changed the offset between $|W| - 1$ to 1. The best results were achieved when $|WS| = 20$, meaning the number of mentions in each window is 20. The optimal number of possible senses was 10, and the optimal offset was 1, as presented in Fig. 3.4. Considering a small number of mentions in each window is because of the nature of our datasets. Three of our datasets were news datasets. Considering these datasets, each item in the dataset includes a small number of words, and mostly all the documents were only one paragraph. So, we decided to consider the number of mentions low in this study. In the continuing studies, as a future direction, we aim to experiment with a higher number of mentions in each window for those datasets coming from Wikipedia.

Our last experiment measures how coherent the whole text is after choosing entity senses with our method compared to the previous method. We calculated the text coherence with the formula introduced in Eq. 3.9. After text disambiguation with our CACW method, the text's coherence is 12% higher on average than the coherence of the same text after disambiguating with Wikisim, both on the short corpus and the long corpus. Here, we report the results of this experiment on the Kore dataset. As mentioned above, this dataset includes 192 mentions. The results of calculating the coherence of the text using our proposed methods and the baseline method Wikisim are provided below in Table 3.5.

	Wikisim	OWCW	CACW
Coherence	0.65	0.75	0.83

Table 3.5: Calculating the text's coherence after disambiguation using the baseline (Wikisim approach) and our two proposed models, CACW and OWCW. We use the coherence metric introduced in Eq. 3.9.

3.5.1 Discussion and Ablation Study

To thoroughly assess the contribution of each component in our proposed models, we conducted an ablation study. The objective was to understand the impact of individual parameters on the model. To achieve this, we performed a series of experiments where we fixed all parameters except one at their optimal values. The optimal values are those obtained from our experiments. Then, we varied the parameter of interest and observed its effect on the outcomes. If we observed substantial variability in the results, we concluded that the particular parameter played an important role in the disambiguation process.

In the first experiment, we focused on the number of possible candidate senses for each mention. We changed the number of senses between one to twenty. In the second experiment, we focused on the window size variable and changed it between five to forty. By fixing all other parameters, we individually examined these two factors' influence, in individual experiments. For each value of the number of possible candidates and the window sizes, we trained and evaluated the model to measure the accuracy. We took the average of the accuracies in each setting and reported this average in Table 3.6. Subsequently, we evaluated the effect of the weighting ranked senses extracted from the CACW algorithm. We used windows without overlap for this experiment and applied a fixed window size throughout the entire input text document. Then, this window overlap was changed from zero to forty, each in a separate experiment. Additionally, we explored the impact of the mention list by changing the number of mentions extracted from the Wikisim baseline from one to fifteen. Each one of these fifteen evaluations was examined separately. If the number of senses for a mention is less than the considered value, we set the algorithm to consider the maximum possible number of senses for those mentions. At the end of each evaluation, the averaged value is reported in Table 3.6.

The evaluation results on the MSNBC dataset are presented in Table 3.6. The table shows that the number of candidate senses and the mention list have a minor impact on the proposed method. Therefore, our algorithm does not rely heavily on them. However,

	Accuracy
Original Settings	0.80
Fixing all parameters except the number of senses	0.79
Fixing all parameters except the window size	0.76
Fixing all parameters except the Offset	0.77
Fixing all parameters except the mention list	0.79

Table 3.6: Ablation Study using MSNBC Dataset. All experiments use the key-entity selection algorithm CACW and are trained with the same epochs. At each experiment, we fix all but one of the variables to evaluate its effect on the model's accuracy. The averaged accuracies are reported.

we notice a clear dependency on the window's size and offset, as the accuracy dropped to 76% and 77%, respectively. It indicates the critical roles of window size and offset in influencing the disambiguation algorithm's performance. Window size and overlap size are more important for capturing the context of a word or phrase, which is essential for Wikification task. If the window size is too small, the model may not be able to capture enough context to identify the correct Wikipedia article accurately. At the same time, if it is too high, the model gets confused. Similarly, if the overlap size is too small, the model may be unable to identify relationships between words and phrases in different windows, leading to errors. The other reason is that the number of mentions and the number of senses are less important for Wikification because they are not as good at discriminating between different Wikipedia articles. For example, two different Wikipedia articles may have the same number of mentions in a document, or two different Wikipedia articles may have the same number of senses. In these cases, window and overlap sizes are more likely to help identify the correct Wikipedia article.

The results show that the number of possible candidate senses for each ambiguous mention has a limited effect on the accuracy. The window size and offset changed the error pattern and corrected some mis-disambiguation errors. This parameter's change resulted in a 4% drop in accuracy. The effect of the mention list on the system's performance is limited, showing that the algorithm does not rely too much on the first step of Wikification, which is mention detection. The comparison between our model and the Wikisim approach shows the importance of the offset parameter and the weighting idea, bringing more context during the disambiguation task.

Regarding the run-time, the similarity calculation should be as fast as possible to be

useful in human interactive processes such as search engines or the inner loop of other computationally intensive algorithms, such as clustering or classification. The complexity of the key-entity modeling is $O(|C|^2)$ when C is the set of all the mentions. In similarities calculation, the key-entity modeling is calculating $2|C|$ similarities, while in our approach, we calculate $|W| \times |C|$, where $|W|$ is the number of windows. This increase in complexity is not dramatic, considering its effect on the accuracy improvement, so this approach still works well to apply for Wikification.

3.6 Conclusion

In this chapter, we worked on the Wikification problem. First, we presented the OWCW (Overlapping Windows Concept Wikifier) algorithm, an unsupervised knowledge-based algorithm for the disambiguation step in the Wikification task. This algorithm finds the correct sense of ambiguous entities based on their context. This algorithm segments the text into overlapping windows in order to consider the context when disambiguating each entity. Second, we proposed CACW (Context-Aware Concept Wikifier) as an extension of the first algorithm to improve the sense disambiguation accuracy by ranking senses based on their similarities to the context. By comparing our Wikification method with other knowledge-based Wikifiers, we showed the validity of our approach in terms of its accuracy for finding correct senses based on the context of the text document. The accuracy ranged between 0.72 for OWCW to 0.93 for CACW. In comparison with the most recent best wikifier RedW, our performance improvement is higher than 5%. We also presented a run-time comparison of our method with a current Wikifier. In summary, our results show that CACW competes with the baseline Wikifiers in terms of complexity while preserving accuracy. We also use a metric to measure the text's coherence after disambiguation, which can also be used in other Wikifiers. This metric shows that the coherence of the text after disambiguation with our proposed algorithm is higher than the value of this metric for the disambiguated texts with previous methods. Analysis of the errors made by our Wikifier and the previous ones shows that our Wikifier corrects the errors raised by the last knowledge-based Wikifiers.

Chapter 4

Classical Machine Learning Models for Categorizing Online Harassment on Twitter

Harassment on social media is challenging since these online platforms are virtual spaces where people express themselves with few restrictions. Furthermore, many users generating publications on online media like Twitter contribute to the difficulty in controlling sexism and sexual harassment, calling for robust methods of Machine Learning (ML) to be applied in this task. This work aims to compare the performance of supervised ML algorithms in categorizing online harassment in Twitter posts and building a baseline. We test Logistic Regression, Gaussian Naïve Bayes, Decision Trees, Random Forests, Linear Support Vector Machines (SVM), Gaussian SVM, Polynomial SVM, and Multi-Layer Perceptron AdaBoost methods on the SIMAH Competition benchmark data using TF-IDF vectors and Word2Vec embeddings as features. As a result, we achieved above 0.80 accuracies for detecting sexism and sexual harassment types in the data. When using TF-IDF vectors, we also show that Linear and Gaussian SVM are the best-tested methods to predict harassment content, while Decision Trees and Random Forests better categorize physical and sexual harassment. Overall, using TF-IDF vectors presented a higher performance on this data, suggesting that the training corpus for Word2Vec negatively influenced the classification task outcomes[1].

This chapter focuses on providing a deep understanding of this dataset, as well as exploring the effectiveness of different classical classifiers for harassment classification, establishing a baseline performance benchmark. This chapter also shows how TF-IDF and Word2Vec might misinterpret the meaning of a sentence due to their limited context awareness, leading to potential misclassifications in harassment detection. We will improve this harassment detection accuracy by using context-aware models in the next chapter, illustrating how the context-aware features can better understand the nuances of language and intent, leading to improved classification accuracy.

[1] Part of this chapter is published in [172].

4.1 Introduction

We begin by exploring the various definitions of sexism. In the Oxford dictionary, in a more general way, it has been defined as "the unfair treatment of people, especially women, because of their sex[2]", while its manifestations on social platforms are different, and it needs to be conceptualized. Sexism in language has been discussed in different communication media, such as advertisements, newspapers, TV, and online social networks. Sexism can be defined as an aggregate of negative stereotypes towards women [206] manifested in language, behaviors, and cultural traits. On social media platforms, sexist comments present different categories according to the intent they are written [189]. It is a widely known problem to automatically detect sexist content online [187, 189, 188]. The data has a significant role when exploring the content on social platforms such as Twitter and employing ML techniques [187]. The reason is that several processes must be done to balance, collect, label, or even measure the overlap of datasets' classes [187].

Detecting harassment on online social media is a challenge in Natural Language Processing (NLP). For instance, Facebook and Twitter were created in 2004 and 2006. Since then, tools to control the spread of offensive posts against women and other groups have been released despite being unable to detect 100% of the content in these networks. The discussion concerning online harassment types, in turn, only gained large audiences in 2017 due to the #MeToo movement, which encourages women to denounce offensive content towards them in real life and on online platforms [188]. Therefore, this is a promising research area [189, 187], and its results have an influential role in promoting an educational culture among social network users, as well as combating online harassment.

In this study, we focus on sexual harassment categorization in tweets. The tweet dataset we use in this study is gathered by Sharifirad et al. [187]. Using NLP and supervised classifiers, in the first task, we classify tweets into two groups: "harassment" versus "no harassment". Considering the harassment types in the dataset introduced by Sharifirad et al. [186], the second task categorizes different types of online harassment tweets into three categories: "indirect harassment", "physical harassment", and "sexual harassment". The machine learning algorithms we use for both tasks include Logistic Regression, Naïve Bayes, Decision Tree, Support Vector Machines, Random Forest, Multilayer Perceptron, and AdaBoost. The necessity of a robust classifier to detect this kind of content justifies

[2]https://www.oxfordlearnersdictionaries.com/definition/english/sexism

testing all of those different methods. The code for this project is publicly available [3].

The remainder of this chapter is organized as follows: Section 4.2 presents related work to the problem. Section 4.3 includes the methodology, the dataset, algorithms, and word embeddings employed in this work. In Section 4.4, we will go through the experiments and their required steps. In Section 4.5, we present the results obtained in our experiments. Finally, Section 4.6 presents the conclusion and future works.

4.2 Related Work

The task of classifying sexism was introduced by Waseem et al. [215]. They collected 16 thousand tweets from a TV show called 'My Kitchen Rules' and annotated them using the hashtag "mkr". Then, they categorized those tweets as racist, sexist, or neither. They used different methods, such as word grams and logistic regression with 10-fold cross-validation, to classify the tweets. A group of researchers classified the tweets that might include sexism, considering the three features "protective paternalism", "complementary gender differentiation", and "heterosexual intimacy" [70]. The deep neural network models used by Badjatiya et al. [6] for hate speech detection in tweets combine long-short term memory (LSTM) with random embedding. Among these models, the one utilizing gradient-boosted decision trees demonstrated the best performance [6].

The importance of the pre-processing step in improving the performance of methods for analyzing Twitter messages has been recognized [18, 82, 48]. Another technique commonly used to enhance harassment detection on Twitter is text augmentation, which involves adding more information to the dataset to address the lack of training data for specific types of harassment, like indirect harassment that is often presented as sarcasm or jokes targeting women [188].This method has been used in different applications like bioinformatics [90], image processing [76], computer vision [30], video and audio processing [50, 107]. However, no text augmentation is performed since our main goal is to measure how robust supervised classifiers are when running on imbalanced, non-structured, and ambiguous textual data.

[3] https://github.com/mozhgans/Competition-Categorizing-Online-Harassment-on-Twitter

4.3 Methodology

This section describes the Twitter harassment dataset, the nine classifiers trained to classify sexist content on social media, and the Word2Vec model [120].

4.3.1 Data

The dataset comprises 10,622 posts collected from Twitter in the English language, with statistics presented in Table 4.1. In this benchmark for harassment content classification, 6,374 data instances are available for training, 2,125 data instances were released for validation, and 2,123 were provided for the test. The number of tweets that present harassment content is 3,956, and sexual harassment is the most numerous hate speech in this set, with 3,419 data instances. Besides sexual, there is also indirect and physical harassment content in the data. The former concerns sexism and indirect offenses, involving conduct that is not directed at a particular individual but results in overall unwanted content, like the invasion of personal space, suggestive remarks or sounds, offensive jokes, ridicule, or innuendo, while the latter refers to violent threats on social media. Analyzing the table shows that the classes are imbalanced. So, special attention is required in the pre-processing step for balancing the dataset classes.

Subset	Data Instances	Harassment Type	Indirect Harassment	Physical Harassment	Sexual Harassment
Training	6,374	2,713	55	76	2,582
Validation	2,125	632	71	36	525
Test	2,123	611	197	100	312
Total	10,622	3,956	323	212	3,419

Table 4.1: Dataset Statistics. The detailed number of tweet posts in each harassment type of our dataset is shown in this table.

The content of the Tweets in this dataset comprises coarse language and swear words, as shown in Fig. 4.1. Each kind of harassment presents specific language constructions. For instance, the words in tweets with indirect harassment content are related to sexism and offensive jokes (Sub-Fig. 4.1b). Physical harassment tweets present violent threats (Sub-Fig. 4.1c), and sexual harassment posts are composed of terms and expressions with a sexual connotation (Sub-Fig. 4.1d).

(a) Harassment content.

(b) Indirect harassment.

(c) Physical harassment.

(d) Sexual harassment.

Figure 4.1: Word clouds for the tweets that present each kind of harassment content.

4.3.2 Algorithms

The contribution of this work is on both binary and multi-class classification tasks. The goal of binary classification is to learn a function $f(x)$ that minimizes the misclassification probability $P(yf(x) < 0)$, where y_x is the class label for x with $+1$ for positive and -1 for negative [128]. Multi-class classification aims at assigning one of the n classes to each input by mapping each input vector x to binary vectors y, where $y \in \{0, 1\}^n$. Since detecting harassment is a relatively new NLP task, we proposed comparing supervised classifiers' performance for this problem. Therefore, different algorithms were chosen to be tested in our study. Among the classification methods, we use the supervised models, as listed below, since the supervised algorithms are more accurate compared to unsupervised algorithms [53].

- **Logistic Regression** [128];
- **Gaussian Naïve Bayes** [128];
- **Decision Trees** (DTs) [128];
- **Linear**, **Gaussian**, and **Polynomial Support Vector Machines** (SVM) [15];

- **Random Forest** (RF) [17];
- **Multilayer Perceptron** (MLP) [128];
- **AdaBoost** [47].

Many effective binary classification methods include kernel, ensemble, and deep learning. Ensemble methods include boosting, random forests (RF), and deep learning based on artificial neural networks (ANNs). In addition, tree-based learning algorithms are considered one of the best and most used supervised learning methods [179].

4.3.3 Feature Extraction

Feature extraction is transforming raw data into numerical features that can be processed while preserving the information in the original data set [136]. This process can be done automatically or manually. The selected features contribute most to the prediction variable or output we are interested in. It often yields better results than applying machine learning directly to the raw data. In NLP, feature extraction and data cleaning are two of those trivial steps that are helpful to understand better the context of what we are dealing with. After the initial text is cleaned and normalized, we need to transform it into its features to be used for modeling. We use specific methods to assign weights to particular words within our document before modeling them. We use numerical representation for individual words as it is easy for the computer to process numbers; in such cases, we use word embeddings.

As discussed, word embeddings are continuous representations of words and their semantic features in low-dimensional vector spaces [24, 65]. It can capture the context of a word in a document, semantic and syntactic similarity, and relation with other words. Word2Vec [120] is one of the most popular techniques to learn word embeddings, whose representations can be obtained using two methods involving neural networks. The first method is common bag-of-words (CBOW), and the second is skip-gram, using hierarchical softmax or negative sampling. CBOW takes the context of each word as the input and tries to predict the word corresponding to the context [24], minimizing the values for the following loss function:

$$\text{loss} = -\log(p(\overrightarrow{w_t}|\overrightarrow{W_t})) \tag{4.1}$$

in which w_t is the target word in the sequence of words W_t (represents the set of context words that surround the target word). In the CBOW model, the sequence of words is transformed into a vector representation by replacing each word in the sequence with its corresponding one-hot encoding. The skip-gram model predicts the surrounding words based on a target word, which is also regarded as the central word [120].

4.4 Experiments

The experiments pipeline starts with the dataset pre-processing, which comprises tokenization, stop-word removal, stemming or lemmatization, and word vector extraction. Then, we test different approaches to develop features for the classifiers, and after that, we perform the classification tasks, validating the models on a 10-fold cross-validation setup.

4.4.1 Pre-processing

This work starts with the pre-processing step, which includes removing hyperlinks, hashtags, numbers, and punctuation marks. For the pre-processing step, we use the functions provided by NLTK library[4]. They are then tokenized with "word-tokenize". Next, the stop-words are removed for the English language. The lemmatization is done with WordNet Lemmatizer [5], or stemming with the SnowBallstemmer [6]. Finally, in addition to the standard English stop words from Scikit-learn[7] package, we remove from the text the Twitter acronyms and HTML tags shown in Table 4.2 since they play a role as noise in the dataset. As we tested two different representations for the words in the tweets, the lemmatized words were used to extract their representations in the Word2Vec model. In contrast, the stemmed forms of the exact words were used to yield their term frequency-inverse document frequency (TF-IDF) [177] representation with the 'TfidfVectorizer' function from Scikit-learn.

Aditional Stop Words
'don', 'http', 'amp', 'cc', 'rt', 'x89', 'x8f', 'x95', 'x9d', 'na', 'im', 'co', 'id'

Table 4.2: Acronyms and HTML tags assigned as stop words in the tweets dataset pre-processing.

[4] https://www.nltk.org
[5] https://www.nltk.org/_modules/nltk/stem/wordnet.html
[6] https://www.nltk.org/_modules/nltk/stem/snowball.html
[7] https://scikit-learn.org/

After that, the TF-IDF vectors were computed for the model to classify. The original dataset (prior training and test sets) had 19,945 words, which led to a large and sparse matrix. Thus, it was necessary to decrease the dimension of this structure to reduce time and computational complexity in the classification. We pruned the number of terms to select the most relevant ones. We started by selecting the 25 most relevant terms according to TF-IDF scores. We increased this number to 50, running the nine supervised classifiers described in Section 4.3.2, with 10-fold cross-validation and measuring accuracy for each model. Accuracy is the ratio of correctly classified instances (regarding the ground truth) to the total number of instances in the dataset. The 45 most relevant words in the data when using TF-IDF scores to extract features from the text are shown in Table 4.3. On the validation set, the achieved accuracies for supervised classifiers while varying the number of features from 25 to 50 with a stride of 5 suggested that 45 is the best number of features extracted from the tweets, as demonstrated in Table 4.4. Using 45 features is the best since it shows the best accuracy results for four of the nine models.

Words Selected According to TF-IDF Scores
‘alway’, ‘ass’, ‘ava’, ‘becaus’, ‘bitch’, ‘black’, ‘chop’, ‘cumshot’, ‘friend’, ‘fuck’, ‘girl’, ‘good’, ‘got’, ‘guy’, ‘horni’, ‘just’, ‘know’, ‘like’, ‘littl’, ‘look’, ‘love’, ‘make’, ‘nake’, ‘nude’, ‘peopl’, ‘porn’, ‘pussi’, ‘realli’, ‘right’, ‘sassi’, ‘say’, ‘sex’, ‘shame’, ‘shit’, ‘slut’, ‘think’, ‘time’, ‘today’, ‘video’, ‘want’, ‘watch’, ‘whore’, ‘women’, ‘xxx’, ‘year’

Table 4.3: The 45 most relevant words in the dataset after stemming with SnowBall stemmer. The words in the table are the direct results of the stemmer.

This dataset is imbalanced, as Table 4.1 shows. It means that the distribution of classes within the dataset is significantly unequal, indicating that one class or category has a much larger number of instances than others. To avoid the major class’s dominant influence on the algorithms’ outcomes, we have employed SMOTE (Synthetic Minority Over-sampling Technique) [29] before running the models. SMOTE makes minor classes equal to major classes by yielding synthetic samples with fewer recurrent labels.

4.4.2 Task A – Binary Classification

According to TF-IDF scores, we test different numbers of features extracted from text, as mentioned in Section 4.4.1. Then, we run the models for each set of features to classify

whether a tweet has harassment content. The accuracy scores for each model on the validation set are presented in Table 4.4. Among the classifiers, LR, RF, Linear SVM, Gaussian SVM, MLP, and AdaBoost reached validation accuracies above 0.90. GNB and DT did not perform as well as the other models; since they are based on probabilities, many words overlap in both classes, leading to lower performance in assigning the correct label to the data points. In other words, this lower performance in assigning the correct labels is attributed to the presence of overlapping words or features in both classes, which affects the classifiers' ability to differentiate between the classes accurately.

Classifier	**20 F.**	**25 F.**	**30 F.**	**35 F.**	**40 F.**	**45 F.**	**50 F.**
Logistic Regression	0.896	0.904	0.905	**0.907**	**0.907**	**0.907**	0.906
Gaussian Naïve Bayes	0.838	0.840	**0.844**	0.842	0.831	0.825	0.822
Decision Trees	0.887	0.888	**0.894**	0.888	0.891	0.888	0.890
Random Forest	0.889	0.894	0.896	0.896	0.899	**0.900**	0.899
Linear SVM	0.896	0.904	0.904	**0.908**	0.906	0.906	0.904
Gaussian SVM	0.887	0.897	0.896	0.901	0.802	**0.903**	0.895
Polynomial SVM	**0.788**	0.702	0.702	0.702	0.702	0.702	0.702
MLP	0.896	0.702	0.903	0.702	**0.906**	0.901	0.903
AdaBoost	0.891	0.903	0.904	0.905	0.904	**0.906**	0.898
Average	**0.874**	0.848	0.872	0.850	0.861	0.871	0.869

Table 4.4: Accuracy values for the supervised methods on the validation set using features extracted by TF-IDF scores. The number of different features (most relevant terms according to TF-IDF scores) varied from 20 to 50, and each one is represented in one column. The highest results of each classifier on each feature are highlighted in bold.

We validate all the models on 10-fold cross-validation before classifying the validation set. In the case of 10-fold cross-validation, the data is divided into 10 equal-sized parts or folds, then training the model on 9 of the folds and evaluating its performance on the remaining fold. This is repeated 10 times using a different fold as the evaluation set, while the other nine folds are used for training.

The learning curves for each model on k-fold cross-validation with $k = \{2, 4, 6, 8, 10\}$ are shown in Fig. 4.2. Linear and Gaussian SVMs achieved the highest scores, whereas GNB and Polynomial SVM achieved the lowest. The remaining classifiers surpassed 85% accuracy.

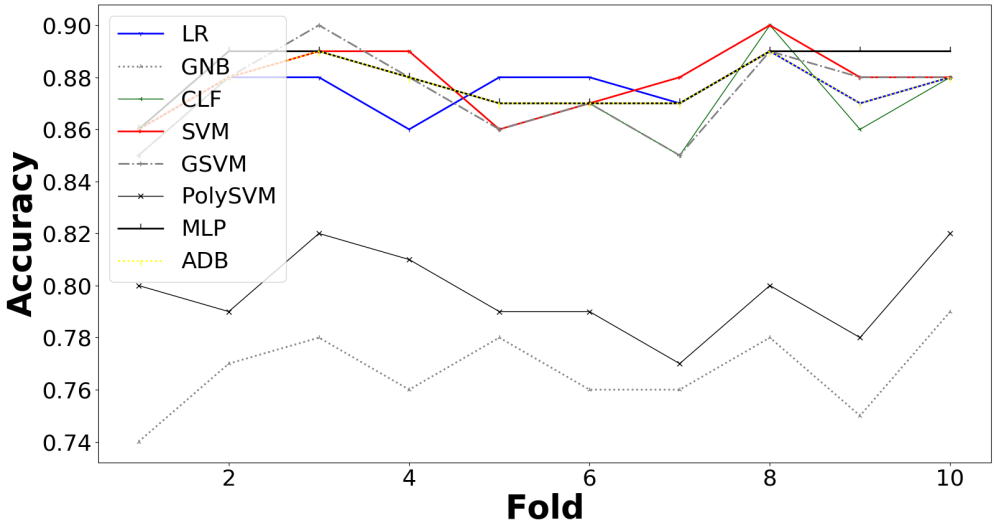


Figure 4.2: Classifiers learning curve on a k-fold cross-validation over the validation set with 45 TF-IDF features.

4.4.3 Task B – Multi-Class Classification

This section concerns the multi-class classification of online harassment in tweets into three categories, namely “indirect harassment”, “sexual harassment”, and “physical harassment”. We run the same algorithms over the dataset using the 45 features extracted in Section 4.4.1. Physical harassment was the most complex content to classify, with 86% average accuracy, since it is the least frequent one among the classes in the dataset. On the other hand, sexual harassment was easier to detect in the tweets, with 94% average accuracy, as shown in Table 4.5, because sexual threats are explicitly written in the posts. RF results surpassed the remaining methods on the three labels, achieving 0.935 of average validation accuracy. The reasons include: 1) Ensemble Learning: Random forests are ensemble learners, meaning they combine multiple decision trees into a single model. This can be particularly effective for complex tasks like harassment classification, where the data may be noisy or non-linear. By combining multiple trees, the random forest can capture a wider range of patterns in the data and reduce the risk of overfitting. 2) Reduced Overfitting: Decision trees are prone to overfitting, which is when they learn the training data too well and fail to generalize to unseen data.

Random forests address overfitting by introducing randomness at two stages: (1) Bootstrap Aggregation (Bagging): Each tree is trained on a random subset of the data with

replacement. This helps to reduce the variance of the forest and prevents any single tree from dominating the prediction. (2) Feature Randomness: At each split in a tree, only a random subset of features is considered. This further reduces the risk of overfitting and helps the forest to learn more generalizable patterns. 3) Robustness to Noise: Random forests are robust to noise in the data. This is because individual trees can make errors, but the overall prediction of the forest is based on the majority vote of these trees. Therefore, errors in individual trees are averaged out, making the forest less susceptible to noise.

Gaussian Naïve Bayes and Polynomial SVM were the worst-performing models, as noted in Section 4.4.2. The reasons for this behavior of Gaussian Naive Bayes include: 1) Oversimplification: Naive Bayes assumes that all features are independent of each other, which is often not the case with real-world data. This can lead to inaccurate predictions, especially for complex tasks like harassment classification. 2) Limited Feature Interaction: Naive Bayes cannot capture complex interactions between features, which can be important for understanding the nuances of harassment language. 3) Sensitivity to Noise: Naive Bayes is sensitive to noise in the data, which can be problematic for harassment data, which may be noisy and contain typos or slang.

The reasons for the behavior of Polynomial SVM include: 1) High Variance: Polynomial SVMs can have high variance, meaning that small changes in the training data can lead to large changes in the model's predictions. This can make them prone to overfitting and unreliable for real-world applications. 2) Computational Cost: Training a polynomial SVM can be computationally expensive, especially for large datasets. This can be a significant disadvantage compared to other models, such as Random Forest. 3) Feature Engineering: Polynomial SVMs require careful feature engineering, which can be time-consuming and difficult. This is because the performance of the model depends heavily on the choice of features and the degree of the polynomial kernel.

To verify how good word embeddings are in detecting harassment content on social media, we trained a 50-dimensional CBOW model on the English Wikipedia corpus, the largest text collection freely available to collect and train models [8]. The results of the validation set are shown in Table 4.6. As this table demonstrates, logistic regression and MLP are the two top accurate classifiers in the binary classification task, the first has 82% accuracy, and the latter is 81% accurate. The reasons for the Logistic Regression results include: 1) Simplicity and Interpretability: LR is a relatively simple model with only a

Classifier	Indirect Harassment	Physical Harassment	Sexual Harassment
Logistic Regression	0.913	0.913	**0.949**
Gaussian Naïve Bayes	0.827	0.807	0.944
Decision Trees	0.899	**0.921**	0.948
Random Forest	**0.923**	**0.931**	**0.952**
Linear SVM	0.912	0.872	**0.949**
Gaussian SVM	0.893	0.854	**0.949**
Polynomial SVM	0.779	0.736	**0.949**
MLP	**0.918**	0.877	**0.949**
AdaBoost	0.913	0.897	0.948
Average	0.886	0.868	**0.949**

Table 4.5: Accuracy values on the three classes of harassment over the validation set using 45 features extracted by TF-IDF scores. The top accurate classifiers for each harassment type are highlighted in bold.

few parameters. This makes it easy to understand how the model works and to interpret its predictions. Additionally, LR can provide insights into the importance of different features through its coefficients. 2) Robustness to Noise: LR is relatively robust to noise in the data, which can be particularly beneficial for TF-IDF features, which can be sensitive to irrelevant words and typos. 3) Good Performance with Linearly Separable Data: When the data is linearly separable, meaning the classes can be separated by a straight line, LR can achieve perfect accuracy. This is often the case for some binary classification tasks where TF-IDF features are used. 4) Computational Efficiency: LR is computationally efficient to train and predict, making it suitable for large datasets and real-time applications.

The reasons for MLP results include: 1) Non-Linear Relationships: MLPs can capture non-linear relationships between features, which can be important for complex tasks like harassment classification. TF-IDF features can capture some non-linear relationships, but MLPs can exploit these relationships more effectively. 2) Feature Representation Learning: MLPs can learn new feature representations from the input data. This can be helpful when the TF-IDF features are not sufficient to capture all the relevant information in the data. 3) High Capacity: MLPs can have a high capacity, meaning they can learn complex models with many parameters. This allows them to fit complex data more accurately. 4) Adaptability: MLPs can be adapted to different types of data and tasks by changing their architecture and hyperparameters. This makes them versatile models that can be used for a wide range of applications.

For categorizing indirect and physical harassment, the Gaussian SVM and random forest are the two top models. Random forest and Gaussian SVM have an accuracy of 97% on average. For categorizing sexual harassment, the best models are random forest, logistic regression, Gaussian NB, and polynomial SVM, and the performance of these models is very close, all about 95%. On average, the best model is AdaBoost with 90% accuracy, and the second best model is random forest with 91% accuracy.

Classifier	**Harassment**	Indirect Harassment	Physical Harassment	Sexual Harassment
Logistic Regression	**0.822**	0.834	0.846	**0.949**
Gaussian Naïve Bayes	0.750	0.834	0.814	**0.949**
Decision Trees	0.706	0.957	0.957	0.939
Random Forest	0.787	**0.963**	**0.973**	**0.950**
Linear SVM	0.731	0.711	0.892	0.948
Gaussian SVM	0.727	**0.966**	**0.982**	0.776
Polynomial SVM	0.822	0.825	0.849	**0.949**
MLP	**0.819**	0.913	0.870	0.813
AdaBoost	0.786	0.935	0.938	0.948
Average	0.772	0.882	**0.902**	0.913

Table 4.6: Accuracy values for the Word2Vec embeddings on the validation set. The top accurate classifiers for each harassment type are highlighted in bold.

4.5 Results and Discussion

The final experiments are performed on the test set, the same data from the challenge proposed by the SIMAH Competition. We compare the performance of TF-IDF feature vectors to Word2Vec embeddings and report the best scoring methods on both sets of features. The results of the test data are presented in this section.

4.5.1 Classification with TF-IDF vectors

We perform both classification tasks on the 45 features extracted by TF-IDF values and measure the accuracy for each run on the test set. LR, Linear SVM, and AdaBoost surpassed the remaining algorithms for binary classification. However, DT and RF achieve the highest results when we run experiments over the three kinds of harassment content. The complete results are shown in Table 4.7.

In order to evaluate the outcomes of the algorithms, we also compute the macro F1 scores for each class, presented in the following table. This performance measure is the harmonic

Classifier	**Harassment**	Indirect Harassment	Physical Harassment	Sexual Harassment
Logistic Regression	**0.814**	0.891	0.834	0.859
Gaussian Naïve Bayes	0.776	0.871	0.818	**0.876**
Decision Trees	0.799	0.879	**0.887**	**0.876**
Random Forest	0.813	0.886	**0.895**	**0.876**
Linear SVM	**0.814**	**0.896**	0.835	0.859
Gaussian SVM	0.810	0.870	0.817	0.859
Polynomial SVM	0.782	0.809	0.759	0.859
MLP	0.811	**0.900**	0.861	0.859
AdaBoost	**0.814**	0.887	0.867	0.862
Average	0.803	0.876	0.841	0.865

Table 4.7: Accuracy values for the TF-IDF vectors. The first column indicates the results of each model on the binary classification task, harassment detection. The next three columns show the results of the models in the multi-class classification task. The last column shows the average accuracy of the classifiers. The top accurate classifiers for each harassment type are highlighted in bold.

mean of precision and recall, and it aims to evaluate the classification performance of all the classes without considering whether the data are balanced. Physical harassment is the hardest kind of offensive content to detect, as is shown in Table 4.8. DT and RF have better physical and sexual harassment classification results, but on average, LR and Linear SVM were the best scoring methods [19]. The results in Table 4.8 are lower than the ones in Table 4.7 since accuracy scores are sensitive to the presence of the major classes [128]. Accuracy is a metric that measures the overall correctness of predictions by comparing the number of correctly classified instances to the total number of instances, which is a commonly used measure in classification tasks [198]. However, class imbalances can influence accuracy, where some classes have a significantly larger number of instances than others. In such cases, accuracy may not provide a complete picture of the model's performance because it tends to be biased toward the major classes. So, we also calculated the F1 score. In Table 4.8, the results are based on the macro F1 score. As mentioned above, the macro F1 score calculates the F1 score for each class independently and then takes the average, giving equal weight to each class. The lower results in Table 4.8 compared to Table 4.7 indicate that when considering the performance of the model on individual classes (as captured by the macro F1 score), it is not performing as well. This suggests that the model may struggle with correctly predicting the minority classes or classes with fewer instances. Thus, the sensitivity of accuracy scores to the presence of major classes might have masked or overshadowed the lower performance

of minority classes in Table 4.7. Therefore, the macro F1 score, which gives equal weight to all classes, provides a more comprehensive evaluation of the model's performance in this scenario.

Classifier	**Harassment**	Indirect Harassment	Physical Harassment	Sexual Harassment
Logistic Regression	0.726	**0.775**	0.598	0.793
Gaussian Naïve Bayes	0.691	0.738	0.593	0.798
Decision Trees	0.697	0.673	**0.613**	**0.802**
Random Forest	0.717	0.686	**0.623**	**0.813**
Linear SVM	**0.728**	**0.773**	0.596	0.793
Gaussian SVM	**0.737**	0.745	0.597	0.793
Polynomial SVM	0.668	0.687	0.568	0.793
MLP	0.726	0.763	0.595	0.793
AdaBoost	0.724	0.726	**0.613**	0.798
Average	0.712	0.729	0.599	0.797

Table 4.8: Macro F1 values for the TF-IDF vectors on each class. The first column indicates the results of each model on the binary classification task, harassment detection. The next three columns show the results of the models in the multi-class classification task. The last column shows the average of the F1 value of the classifiers. The higher accuracies in each harassment type are in bold.

4.5.2 Classification with Word Embeddings

When we run the supervised methods on the dataset represented by the word vectors yielded by the Word2Vec model, the accuracy values for the binary classification are lower than the ones on the TF-IDF features, as Table 4.9 shows. The main reason for this difference in performance is the nature of the training data input to the Word2Vec model. As Wikipedia generally presents only formal language, the swear words in the tweets do not have meaningful representations in the model. Furthermore, if we had trained Word2Vec on the data used in this study, the word representations could lead to overfitting. Among the supervised algorithms, RF and Gaussian SVM had the best results on most tries.

By computing the macro F1 scores (Table 4.10), we can see that LR and Polynomial SVM present the highest average F1 score values as the top scores in most classes. Some possible reasons for this behavior of these two models could be, (1) LR and Polynomial SVM are both linear models. Linear models are relatively simple and easy to train, but they can be very effective for classification tasks. In particular, they are well-suited for tasks where the data is relatively well-structured and there are a large number of features. (2) LR

Classifier	**Harassment**	Indirect Harassment	Physical Harassment	Sexual Harassment
Logistic Regression	**0.781**	0.837	0.822	0.858
Gaussian Naïve Bayes	0.735	0.831	0.809	**0.860**
Decision Trees	0.660	0.899	0.933	0.857
Random Forest	0.760	**0.909**	**0.948**	0.858
Linear SVM	0.727	0.894	0.847	**0.859**
Gaussian SVM	0.707	**0.907**	**0.952**	0.847
Polynomial SVM	**0.781**	0.843	0.836	**0.859**
MLP	0.768	0.869	0.848	0.669
AdaBoost	0.752	0.899	0.912	0.857
Average	0.741	0.876	0.879	0.836

Table 4.9: Accuracy values for Word2Vec Embeddings. The first column indicates the results of each model on the binary classification task, harassment detection. The next three columns show the results of the models in the multi-class classification task. The last column shows the average accuracy of the classifiers. The higher accuracies in each harassment type are in bold.

and Polynomial SVM are both able to learn complex relationships between features. This is important for harassment detection, as harassment can be expressed in a variety of ways, using different words and phrases. Concerning the harassment content in the tweets, sexual harassment was the easiest to detect, primarily due to its higher prevalence in the dataset.

Classifier	**Harassment**	Indirect Harassment	Physical Harassment	Sexual Harassment
Logistic Regression	**0.731**	0.666	**0.599**	**0.791**
Gaussian Naïve Bayes	0.692	0.665	0.589	0.794
Decision Trees	0.608	0.579	0.575	0.782
Random Forest	0.681	0.527	0.528	0.785
Linear SVM	0.533	**0.667**	0.595	0.793
Gaussian SVM	0.414	0.475	0.487	0.458
Polynomial SVM	**0.729**	**0.670**	**0.604**	0.793
MLP	0.716	0.513	0.523	0.583
AdaBoost	0.697	0.652	0.589	0.787
Average	0.644	0.601	0.565	0.729

Table 4.10: Macro F1 values when using Word2Vec Embeddings. The first column indicates the results of each model on the binary classification task, harassment detection. The next three columns show the results of the models in the multi-class classification task. The last column shows the average F1 score of the classifiers. The highest accuracy is in bold.

To assure the performance of our approaches for harassment content classification on social media, we compute the average accuracies for the three kinds of offensive tweets using the scores from Table 4.7 and compare them to the results reported by Sharifirad et al. [187].

The four lines at the top and the bottom of Table 4.11 were extracted from their work. In the top 4 rows of the table, AWR refers to "All Words Replacement", a text augmentation technique [186]. In our experiments, we did not perform this kind of data enrichment. Our results on the 45 TF-IDF vectors surpassed the results reported by Sharifirad et al. [187], without text augmentation. However, our approaches must be improved compared to their augmented dataset results. The rise in accuracy compared to the literature results that do not use text augmentation shows that pre-processing tasks considerably influence the results. Nevertheless, our approach is stable and robust to detect harassment content on Twitter.

Classifier	Average
CNN + AWR	**0.980**
LSTM + AWR	**0.980**
SVM + AWR	0.920
Naïve Bayes + AWR	0.940
Logistic Regression	0.861
Gaussian Naïve Bayes	0.855
Decision Trees	0.880
Random Forest	0.885
Linear SVM	0.863
Gaussian SVM	0.848
Polynomial SVM	0.809
MLP	0.873
AdaBoost	0.872
CNN	0.750
LSTM	0.740
SVM	0.680
Naïve Bayes	0.600

Table 4.11: Average accuracy values for detecting harassment content (using scores from Table 4.7), compared to the results in the literature for detecting harassment, using the same tweet dataset introduced in this chapter. The highest accuracy is in bold.

4.6 Conclusion and Future Works

This chapter aims to improve the classification performance of different models on Twitter data. By considering different types of online harassment on social media, we used nine supervised algorithms to categorize this content. We also empirically compared TF-IDF feature vectors and Word2Vec embeddings trained on the Wikipedia English corpus. Finally,

we validated all the executions of the algorithms on a 10-fold cross-validation process, applied them over a validation set, and then classified the posts collected from Twitter.

Among the nine models to categorize offensive content, DT, RF, and Linear SVM showed the best results. DT and RF classified instances according to information gain, whereas Linear SVM found the hyperplane that maximizes the classes' boundary decision. In social media content, the information gained from the first two algorithms is influenced by the frequency of each word associated with some label. At the same time, in the last one, the hyperplane is calculated according to optimization functions (see [15] for more details). We noticed that embeddings trained on textual corpus whose domain differs from the target data tend to decrease the classification performance. It also showed that these representations are not robust enough to perform well regardless of the data domain over which the predictions are performed. To indicate some of the reasons for our observations based on our experiments, we can point to, (1) DTs and RFs are able to learn complex relationships between features without overfitting. TF-IDF and Word2Vec embeddings are both high-dimensional feature representations, so DTs and RFs are likely to perform well on tasks that use these representations. (2) The superior performance of DTs and RFs is also due to the fact that they are ensemble methods. Ensemble methods combine the predictions of multiple individual learners to produce a final prediction. This can help to reduce overfitting and improve classification performance. (3) Linear SVMs are also well-suited for classification tasks, but they are more sensitive to the choice of hyperparameters. However, since we used a 10-fold cross-validation process to tune the hyperparameters of the linear SVM model, it likely helped to improve its performance. (4) Embeddings trained on a textual corpus with a different domain from the target data may not be as effective for classification. This is because the embeddings may not be able to capture the nuances of the language used in the target domain. For example, if we trained embeddings on a corpus of news articles and then used them to classify tweets, the embeddings may not be able to capture the informal language and slang that are commonly used in tweets.

Some of the Word2Vec Features include: 1) Capturing Semantic Similarity: Word2Vec represents words as vectors in a continuous space, where semantically similar words are closer together. This allows the model to capture the meaning and context of words more effectively than TF-IDF, which simply counts the frequency of words. 2) Efficient Feature Representation: Word2Vec features are dense and high-dimensional, which can be more

effective for capturing complex relationships between words compared to sparse TF-IDF features. 3) Less Prone to Noise: Word2Vec features are less sensitive to irrelevant words and typos compared to TF-IDF, which can be particularly beneficial for noisy data.

Some of the TF-IDF Features include: 1) Simple and Interpretable: TF-IDF features are directly interpretable, as they represent the frequency of words in the documents. This makes it easier to understand how the model is making predictions. 2) Robust to Out-of-Vocabulary Words: TF-IDF features are robust to out-of-vocabulary words, which can be a problem for Word2Vec, as it requires words to be pre-trained in the vocabulary. 3) Effective for Text Classification: TF-IDF has been proven effective for various text classification tasks, including sentiment analysis and topic modeling.

Therefore, the different performance of Word2vec and TF-IDF models can be attributed to the following reasons: 1) Feature characteristics: Word2Vec captures semantic similarity more effectively than TF-IDF, while TF-IDF is simple and interpretable. 2) Model strengths: Random forests are robust to noise and excel at complex non-linear relationships, while LR and MLP perform well with linearly separable data and are computationally efficient. 3) Task complexity: The complexity of the binary classification task can influence the effectiveness of different feature types and models. In conclusion, the optimal choice of model and feature representation depends on the specific characteristics of the data and the desired outcome. Word2Vec may be preferable when capturing semantic similarity is crucial and the data is complex, while TF-IDF may be better for simpler tasks and when interpretability is important. Random forests can be a strong choice when dealing with noisy data and complex relationships, while LR and MLP can be efficient options for linearly separable data.

When we have automatic approaches to detect the various types of harassment on social media, it makes it more applicable to prevent this type of behavior from growing. We also expect this work to leverage the discussions on harassment detection on Twitter and other social networks. Furthermore, in future work, we plan to test deep learning architectures on our set of features and different test strategies of data augmentation.

Chapter 5

Graph Convolutional Network for Categorizing Online Harassment on Twitter

Twitter is one of the social media platforms where people express themselves freely. Harassment is one of the consequences of these platforms, which is hard to prevent. Text categorization and classification is a task that aims to solve this problem. Many studies applied classical machine learning methods and recent deep neural networks to categorize text. However, only a few studies have explored the effectiveness of graph convolutional neural networks simultaneously using classical approaches to categorize harassment in tweets.

In the previous chapter, Chapter 4, we studied the problem of harassment categorization using classical machine learning models. Our experimental results indicate there is still room to improve our categorization accuracy. Harassment categorization is a problem that can be modeled as a relational data problem, where the data points are text messages, and the relationships between the data points are the interactions between the users who sent and received the messages. Graph convolutional models are well-suited for modeling relational data. They can learn from the relationships between the data points, which can help them better understand the context of the messages and identify harassment more accurately. In this chapter, we study the same problem as in the previous chapter while using a graph convolutional model.

In this work, we propose using Graph Convolutional Networks (GCN) for the tweet categorization task along with different embedding models. We use BERT, SBERT, and our representation learning model, C-KASE, which we presented in Chapter 2. Second, we explore this categorization task by using classical machine learning approaches, as in Chapter 4, and compare the results with the GCN model. Third, we show the effectiveness of the GCN model in performing this task by feeding half of the dataset to the model and still obtaining good performance, above 91% for categorizing all different harassment types. Furthermore, we employ various embedding approaches to discover the optimal

representation of harassment within each of the models [1].

5.1 Introduction

Text categorization or text classification is a primary critical task in Natural Language Processing (NLP). Online platforms are attracting more users [189], and people, specifically on social media, tend to express themselves freely. As discussed in Chapter 4, sexism in language has been discussed in different communication media, such as Twitter and other social networks. So, it is a known problem to detect sexist content online automatically. One possible way is to explore content on social platforms, such as Twitter, employing Machine Learning (ML) techniques [60]. The results of this type of categorization problem play a crucial role in fostering an educational culture among social network users and addressing the issue of online harassment.

Text representation is one of the critical steps for text classification. Traditional methods used hand-crafted features like bag-of-words and n-grams. Recently, with the advent of deep learning models such as convolutional neural networks (CNN) [92, 89], and recurrent neural networks (RNN) [184], as well as graph neural networks (GNN) [182, 81], there has been a shift towards utilizing these models for text representation. Between them, graph neural networks are particularly effective in capturing the structural information present in a graph and generating graph embeddings. In the context of text classification, graph-based representations can be constructed by treating text documents as nodes and capturing relationships such as co-occurrence or syntactic dependencies as edges. By leveraging the structural information encoded in these graphs, graph neural networks excel at extracting meaningful features from text data. This property makes them widely used in various applications, including text classification tasks [224]. This property is also referred to as long-range dependencies, meaning GCNs can learn long-range dependencies. In harassment categorization, it is important to be able to capture long-range dependencies in the data. This means that it is important to be able to understand the context of a message, even if it is several messages away from the current message. GCNs can learn long-range dependencies in the data, which can help them better understand the context of the messages and identify harassment more accurately. Another property is that GCNs are good at modeling relationships between data points. Harassment categorization can be thought of

[1] Part of this chapter is published in [169].

as a problem of modeling relationships between data points, where the data points are text messages and the relationships are the interactions between the users who sent and received the messages. GCNs can learn from the relationships between the data points, which can help them to better understand the context of the messages and to identify harassment more accurately.

This study uses a sexual harassment tweets dataset that includes comments in English. This dataset is the same as was used in Chapter 4. Each tweet has a set of labels that shows if the tweet includes harassment or not and which type of harassment is in each tweet; three categories are considered here: "indirect harassment", "physical harassment", and "sexual harassment". We build a graph convolutional network to categorize comments into these three categories and use classical ML approaches to compare classification results. In the first step, we classify tweets into two groups; harassment versus no harassment. The second step categorizes different online harassment tweets into the three mentioned classes. The ML algorithms we use for both tasks include Logistic Regression (LR), Naïve Bayes (NB), Decision Tree (DT), Support Vector Machines (SVM), Random Forest (RF), Multilayer Perceptron (MLP), and AdaBoost. The code for this project is publicly available [2].

5.2 Background

Most previous works in text classification mainly focus on feature engineering and classification algorithms [98, 135]. In most of these studies, the most used feature is the bag of words feature, and later the n-grams feature [224]. Some of these works have converted text to graphs and performed feature engineering on graphs and subgraphs [224, 68]. There are studies showing representation learning method applied in the deep learning algorithm has a significant role [211, 224]. So, in this chapter, we use a graph neural network approach and compare its results with our previous results provided in Chapter 4.

Some of the previous works have used CNN for sentence classification [31] due to their ability to capture local dependencies and hierarchical structures within sentences. These models leverage convolutional operations to extract relevant features, enabling them to accurately classify sentences based on their semantic meaning. Character level CNNs are designed and achieved promising results on text data [227, 33]. Later, in RNN models, by introducing the attention mechanism, these types of classifier models show significant

[2] https://github.com/mozhgans/Classification_harassment

improvement in results [222, 214]. RNNs, by their sequential nature, process information in a step-by-step manner, considering only the context of the current input and the hidden state from previous inputs. This sequential processing may limit their ability to capture long-range dependencies or relationships that exist across the entire corpus. In other words, RNNs lack the capability to incorporate knowledge about how words or elements within a text corpus co-occur or relate to each other in a broader context. This limitation can affect their performance in tasks that require understanding global patterns or making decisions based on a comprehensive understanding of the corpus. To address this drawback, alternative models or approaches, such as transformer-based architectures, have been developed to explicitly capture global dependencies and improve the representation of information within a corpus [126].

Graph neural networks (GNNs) have received growing attention recently in the task of text classification [224]. The graph convolutional model was introduced by Kipf and Welling [81], which shows the best results in many different tasks and benchmarks [65, 81]. The idea of GNNs in text classification is to leverage the underlying graph structure inherent in text data for improved classification performance. The graph represents the relationships between textual elements such as words, sentences, or documents in this context. GNNs can capture the structural information in the graph, allowing them to learn representations that encode local and global dependencies among the text elements. By considering the relationships and interactions between words or sentences within the graph, GNNs can effectively model the contextual information and capture semantic connections. Since text data often exhibit complex relationships and dependencies, by utilizing GNNs, text classification models can benefit from the enhanced representation and understanding of the interdependencies between textual elements, leading to improved performance in tasks such as text classification. GNNs in text classification offer a powerful framework to exploit the inherent graph structure in text data, enabling more accurate and nuanced modeling of the relationships between textual elements and, consequently, enhancing the performance of text classification tasks [126, 221].

5.3 Tweet Categorization

Our tweet categorization system aims to perform binary and multi-class classification, similar to the goal of the previous chapter. In binary classification, our goal is to categorize

the data instances into one of two groups. In multi-class classification, the attempt is to assign each data instance to one of several classes. In the following section, we will introduce our method in detail. First, we show how to build the graph convolutional network, a powerful neural network designed to work directly on graphs and leverage their structural information. Then, we show how information is propagated through the hidden layers of a GCN. Next, we see how the GCN aggregates information from the previous layers and how this mechanism produces useful feature representations of nodes in graphs. Then, we explain how to predict the label for a given text based on the learned representations. Finally, we provide information about the supervised classical ML approaches, use them for this tweet categorization, and compare the results of all models together.

Graph Convolutional Network

The goal of Graph Convolutional Networks (GCNs), in general, is to perform machine learning tasks on graph-structured data [81]. GCNs aim to learn node representations by considering both local and global information within the graph. GCNs can extract features and make predictions on graph-structured data by capturing the relationships and dependencies between nodes. In the context of text classification, the goal of GCN is to leverage the underlying graph structure present in text data and enhance the classification performance. The graph can represent relationships between words, sentences, or documents. So, GCNs can capture the contextual information and dependencies among text elements. This allows GCNs to learn representations that encode both local and global information, enabling them to classify text data into different categories or classes effectively [226, 204]. GCNs have a multi-layer neural network architecture, which is mostly used when the machine learning algorithm will be applied on graphs [81], meaning GCN operates directly on a graph.

For the rest of this subsection, we built our model based on the formal notations introduced in the original GCN paper [81]. We state the original GCN model before mapping it to our problem. In the settings following the original GCN work, the graph is depicted as $G = (V, E)$, where $V(|V| = n)$ is the set of the nodes, and E is the set of the edges, similar to the GCN original paper [81, 224]. Every node is assumed to be connected to itself, i.e., $(v, v) \in E$ for any v. The reason for this assumption is mentioned at the end of this paragraph. $X \in R^{n \times m}$ is a matrix containing all n nodes with their features, where

m is the dimension of the feature vectors. Each row $x_v \in R_m$ is the feature vector of a vertex v. Similar to previous work [204, 170], we consider an adjacency matrix A of G and its degree matrix D, where $D_{ii} = \sum_j A_{ij}$. Because of self-loops, the diagonal elements of A are all 1. We now have a graph, its adjacency matrix A, and a matrix of input features X. After applying the propagation rule $f(X, A) = AX$ and $X = I$ (the identity matrix is used as a component to preserve the existing node features during the convolutional operation), the representation of each node (each row) is now a sum of its neighbor's features. In other words, the graph convolutional layer represents each node as an aggregate of its neighborhood. Considering the self-loops in the graph is because of the aggregated representation of a node to include its features means including the identity matrix in the propagation rule of GCN ensures that the node features are preserved and incorporated during the information propagation process, enabling the model to leverage both local and individual node information for improved performance. The architecture of this GCN is inspired by Japsen in the paper with the title "How to do Deep Learning on Graphs, with Graph Convolutional Networks" [69].

For a one-layer GCN, the new k-dimensional node feature matrix $L^{(1)} \in R^{n \times k}$ is computed as the following formula, which is introduced by Kipfl et al. [81, 224],

$$L^{(1)} = \rho(\hat{A}XW_0), \tag{5.1}$$

where $\hat{A}$ is $D^{-0.5}AD^{-0.5}$, the normalized symmetric adjacency matrix and $W_0 \in R^{m \times k}$ is the weight matrix. The ρ is the activation function (RELU); $\rho(x) = max(0, x)$. The weights in the weight matrix are typically learned through the training process. This weight matrix is a learnable parameter matrix in GCN. It determines how the node features are transformed during the convolutional operation. The dimensions of the weight matrix depend on the input and output feature dimensions. If the input node feature matrix has dimensionality N $\times$ D, where N is the number of nodes and D is the dimension of each node's feature, and the desired output dimension is K, then the weight matrix W will have dimensions D $\times$ K. During the training phase, the weights in the weight matrix W are optimized to minimize a predefined loss function. This optimization process involves iteratively adjusting the values of the weights based on the gradients computed through backpropagation. The specific optimization algorithm, such as stochastic gradient descent (SGD) or Adam, is typically used to update the weights and improve the performance of the GCN model on the

given task. By learning the weights in the weight matrix, the GCN model can adaptively determine the importance and transformation of each feature dimension for the target task. This flexibility allows the model to effectively capture and utilize relevant information in the node features, enhancing its capability for tasks such as node classification, link prediction, or graph-level prediction.

When multiple GCN layers are stacked, information about more prominent neighbors is integrated using this update rule;

$$L^{(j+1)} = \rho(\hat{A}L^{j}W_{j}), \tag{5.2}$$

where j is the layer number and $L^0 = X$. In other words, the size of the second dimension of the weight matrix determines the number of features at the next layer. The feature representations can be normalized by node degree by transforming the adjacency matrix A by multiplying it with the inverse degree matrix D. First, we use the simple propagation rule $f(X, A) = D^{-1}AX$, then improve it. The improved version is inspired by recent works [81, 224] that proposes a fast approximate spectral graph convolution using a spectral propagation rule $f(X, A) = \sigma(D^{-0.5}AD^{-0.5}XW)$. They show this property is very useful, that connected nodes tend to be similar, e.g., have the same label.

We consider each tweet post as one node of the graph, and a newly added node will connect with its nearest neighbor using cosine similarity, which makes the edges of the graph. The cosine similarity between two nodes on the edges makes the weight matrix. We start creating the graph by selecting five nodes from our corpus; three of them from the three different harassment classes, one node as the tweet we know contains harassment and one node that is not harassed. In each category, we define a representative node for that class, the average representation of all nodes. In the first step, each class representative and the nodes are the same. The averaging process for calculating the representative node starts from the second step when we add new nodes to the graph to categorize. The dependencies between nodes that represent similarities are weights on the edges. The number of nodes in the text graph $|V|$ is the number of tweets (corpus size). We set the feature matrix X as an extracted representation of SBERT [3] as input to GCN.

For the representation, we test different embedding representations including Glove [151], TF-IDF [177], BERT [38], SBERT [163], LMMs [105], and C-KASE. It is important to

[3] Our settings are based on https://www.sbert.net

note that when employing word-based embedding models, we extracted embeddings for individual words in each tweet and employed an averaging technique for aggregation. We then used the resulting vector as the representation of the entire tweet. Other aggregation methods are also possible to consider here. In the result section, we reported the result of the winner–in terms of accuracy– which is SBERT.

As mentioned, the weights of the edge between node i and node j are defined as:

$$W_{ij} = \text{cosine sim}(R(i), R(j)) = \frac{R(i) \cdot R(j)}{||R(i)||||R(j)||} \tag{5.3}$$

where $R(i)$ is the representation of node i.

We pass the graph after building it into a simple 2-layer GCN, as in GCN and Text GCN [81, 224], the second layer node (tweet) embeddings are fed into a softmax classifier:

$$Z = softmax(\hat{A}RELU(\hat{A}XW_0)W_1) \tag{5.4}$$

where

$$\hat{A} = D^{-0.5}AD^{-0.5} \tag{5.5}$$

and

$$softmax(x_i) = \frac{1}{Z}exp(x_i) \tag{5.6}$$

with $Z = \sum_i exp(x_i)$.

For the loss function, we also follow the loss function that Kipfl et al. defined in their work as $L = -\sum_{d \in Y_D} \sum_{f=1}^{F} Y_{df} ln Z_{df}$ [81]. In this loss function definition, Y_D is the set of tweet indices that have labels, and F is the dimension of the output feature [224]. Y is the label indicator matrix. The weight parameters W_0 and W_1 can be trained via gradient descent, similar to Text GCN [224]. The matrix $\hat{A}XW_0$ contains the first layer tweets and embeddings, and $\hat{A}RELU(\hat{A}XW_0)W_1$ contains the second layer tweets and embeddings. This architecture helps to pass information between a pair of nodes (pair of tweets). This 2-layer GCN performs message passing between nodes to a maximum of two steps away. This GCN model on this Tweets dataset performs better than a one-layer model and models

with more than two layers, based on recent works [81, 97, 66]. Using a 2-layer GCN instead of a single layer can provide several advantages and improvements in capturing more complex relationships and higher-order dependencies in the graph-based data. One reason for using multiple layers in GCNs is to capture Local and global information; a 2-layer GCN allows the model to capture both local and global information from the graph. The first layer aggregates information from the immediate neighbors of each node (tweets) in the graph, capturing local dependencies. The second layer then aggregates information from the first layer's representations, allowing the model to capture more global dependencies and interactions across the entire graph. Our model's validity is demonstrated through experimental performance comparisons with recent models [81, 97], in Section 5.4.

5.4 Experiments

We evaluate the accuracy of the classical ML and GCN models on the tweets categorization task. Later, we assess if our GCN model works well even with a limited number of tweets.

5.4.1 Classical Machine Learning Algorithms

We compared the performance of supervised classifiers for this problem with our GCN model because harassment detection was a relatively brand-new NLP task (at the time of this work, December 2020). For this comparison, we applied different supervised classifier models, as discussed in Chapter 4 in detail. For example, Kernel, ensemble, and deep learning methods are common binary classification methods [101]. We chose the following classifiers based on our recent work [172, 218], Logistic regression, Gaussian Naïve Bayes, Decision Trees (DTs), linear, Gaussian, and polynomial Support Vector Machines (SVM). The word embeddings were those that we used in the previous chapter. We also tried BERT, Sentence BERT (SBERT), and C-KASE, our new proposed embedding model introduced in the second chapter.

5.4.2 Baselines

We compare our GCN model with multiple state-of-the-art text classification and embedding methods as follows:

- TF-IDF + Classical ML models

- Word2Vec + Classical ML models [172]
- CNN [79]
- LSTM and Bi-LSTM [101]
- fastText [74]

5.4.3 Data

This dataset is the same dataset introduced in the previous chapter, the SIMAH's competition dataset [4]. This English Twitter harassment dataset comprises $10,622$ posts collected from Twitter in English. The data statistics are presented in Table 4.1. In this dataset, the number of data instances for training is $6,374$. The number of data instances for validation is $2,125$. Finally, the number of data instances for the test is $2,123$. The statistics show that sexual harassment is the most numerous kind of hate speech in this set with $3,419$ data instances, while the number of tweets including harassment content is $3,956$. The second type of harassment available in data points is indirect harassment. This type pertains to sexism and indirect offenses that do not target a specific individual but contribute to an overall toxic environment. The third type of harassment available in this dataset is physical harassment, which refers to violent threats on social media. We also observe from the dataset summary table, Table 4.1, that classes in our data are not balanced. So, special attention is required in the pre-processing step for balancing the dataset classes.

5.4.4 Settings

In this section, we describe our experimental setup. First, we started by pre-processing the data. This includes tokenization, removing stop-words, lemmatization, and word representation extraction[5]. The pre-processing task for this Tweet dataset includes removing hyperlinks, hashtags, numbers, and punctuation marks. We removed the Twitter acronyms and HTML tags, including [don, http, amp, cc, rt, x89, x95, x9d, na, im, co, id].

After pre-processing, we performed the classification tasks and validated the models on a 10-fold cross-validation setup. As we tested different representations for the tweets, the

[4] You can request the data from the SIMAH's competition website: https://competitions.codalab.org/competitions/?q=simah

[5] We used the provided functions by NLTK library `https://www.nltk.org/`.

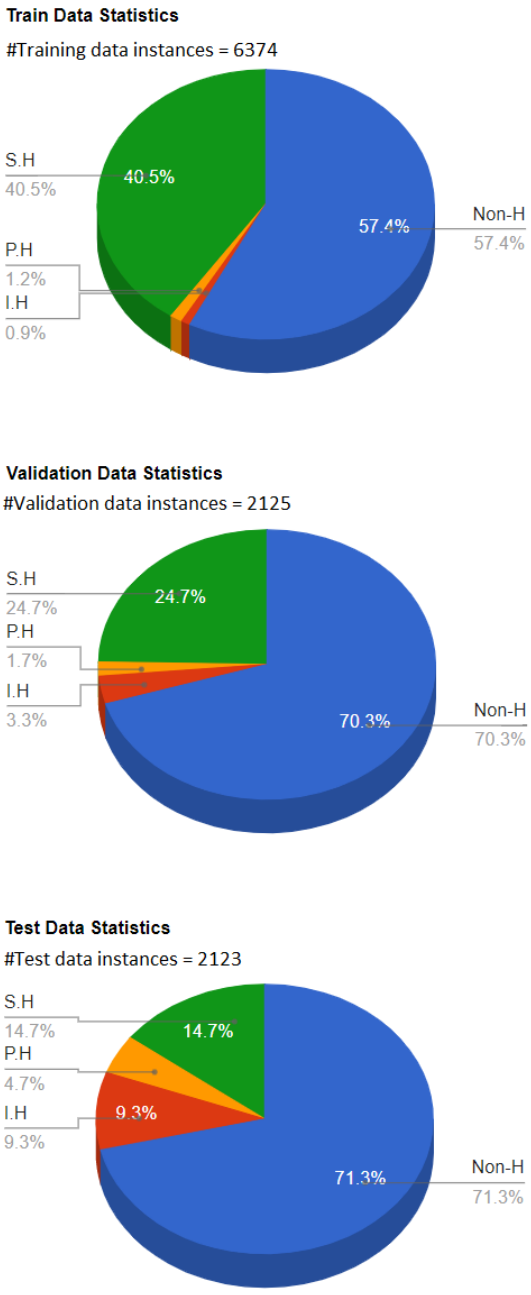


Figure 5.1: Dataset Statistics. 'Indirect H.' stands for indirect harassment, 'Physical H.' refers to physical harassment, and 'Sexual H.' means sexual harassment. The first figure shows the statistics of each harassment class in the training data. The second figure represents the statistics of the validation dataset and number of instances in each class. The last figure represents the statistics and number of instances in each class of the test dataset.

lemmatized words were used to extract their representations in the Word2Vec, whereas the stemmed forms of the exact words were used to yield their term frequency-inverse document frequency (TF-IDF) representation with the 'Tf-idf Vectorizer' function from Scikit-learn. We run experiments with the classical classifiers [172] for reducing the sparsity of the representation matrix of TF-IDF scores, and SMOTE for balancing the dataset[6]. SMOTE generates synthetic samples for the minority class by interpolating between neighboring instances, thus increasing the presence of underrepresented classes. For GCN and the Tweets representation with sentence transformers, we used Sentence-Bert[7]. Following [81, 224], we randomly selected 10% of the training set as a validation set and trained GCN for a maximum of 200 epochs using Adam [80] and stopped training if the validation loss does not decrease for 10 consecutive epochs.

5.5 Results and Discussion

The accuracy score of each model is presented in Table 5.1. The GCN model outperforms other models in the first binary classification task, including the classical classifiers. The TF-IDF+RF can outperform CNN with randomly initialized word embeddings. As we observed from the results, LR, RF, Linear SVM, Gaussian SVM, MLP, and AdaBoost have shown better performance in comparison with GNB and DT. Polynomial SVM with degree 2 has also not performed well due to the likelihood of overfitting caused by the polynomial kernel, as shown in Table 5.1. The polynomial kernel is capable of capturing complex relationships between features by mapping them to a higher-dimensional space. However, when the degree of the polynomial is set too high, such as in degree 2, it can lead to overfitting. Overfitting occurs when the model becomes too specific to the training data and fails to generalize well to unseen data. In the case of Polynomial SVM with degree 2, the model may be overly flexible and fit the training data too closely, including noise or irrelevant patterns. As a result, its performance on new, unseen data, such as tweets in the classification task, can suffer.

For the task of multi-class classification of online harassment tweets into three categories of "indirect harassment", "sexual harassment", and "physical harassment", we run the same algorithms over the dataset. Physical harassment is the most challenging content

[6] SMOTE[29] for balancing the classes.
[7] https://www.sbert.net

to classify since it is the least frequent among the dataset's classes. On the other hand, sexual harassment is easier to detect in tweets because sexual tweets are explicitly written in the posts. RF results surpass the remaining methods on the three labels, hitting 0.935 of average accuracy on the test data. The binary classification task shows that Gaussian Naïve Bayes and Polynomial SVM are the worst-performing models. The GCN model introduced in this work performs better than all classical classifiers. All of these results are shown in Table 5.1. The reason for obtaining good accuracies from the GCN model in comparison with classical ML approaches relates to the fact that GCN models capture local and global dependencies, meaning they consider the graph structure and model interactions between neighboring words, enabling them to better represent the local and global context of the text. When we use the GCN model with contextual word embeddings, such as those generated by pre-trained models like BERT, the model captures the meaning of words based on their surrounding context. These embeddings provide rich and informative representations, improving the quality of feature vectors used for classification. For tasks that involve document-level or paragraph-level relationships, GCNs can integrate graph-based information naturally. The GCN model captures not only word-word relationships but also sentence-sentence or document-document interactions, providing more context-aware and informative representations for classification. In the case of GCN with SBERT, sentence embeddings from SBERT are used as the initial node representations for the graph. These sentence embeddings have been specifically designed to capture the semantic meaning of entire sentences, making them potentially more suitable for this certain tweet classification task. While BERT and C-KASE provide contextual embeddings at the word level, they do not explicitly capture sentence-level relationships in the graph. The GCN may still capture some contextual information, but the sentence-level relationships might not be as prominent. SBERT focuses on generating semantically meaningful sentence embeddings, which inherently capture sentence-level relationships. This sentence embedding model performs better in our categorization problem since our data is a collection of tweets, and almost all the tweets are in the form of full sentences rather than words. When the task is classifying the words, the word-level embeddings, such as BERT and C-KASE, perform better than SBERT. By using SBERT embeddings as input to the GCN, the model can more explicitly leverage sentence-level meaning, which can be advantageous for our text classification task that heavily relies on understanding the entire sentence's semantics. SBERT uses a weighted

combination of mean and max pooling to create sentence embeddings. Mean pooling is a simple method that averages the word embeddings for all the words in a sentence. Max pooling is a more aggressive method that takes the maximum value of the word embeddings for all the words in a sentence. The training data determine the weights. This means that SBERT is able to learn which pooling strategy is more effective for a particular task.

The information provided in Table 5.1 is also demonstrated as a heatmap in Fig 5.2, as a visualization tool. This heatmap is a visual tool, allowing readers to easily interpret and compare accuracy differences among the tested models.

In some NLP tasks, a critical limitation for the models is the dependency of the model on the amount of annotated corpora [75]. Therefore, building a model that can learn from a limited set of examples is crucial. In one of our experiments, we used half of the randomly selected data points in each class and evaluated the GCN model for this aim. The model's performance is shown in Table 5.2 when the data we use is complete versus when it is halved. The performance drops only %3. Compared with the accuracy of other models, even on the complete dataset, this result shows the robustness of the outcomes of the GCN model.

5.6 Conclusion

In this chapter, we aimed to improve the classification performance of different models on a collection of Twitter data, including classical machine learning and deep neural network models. Besides using classical machine learning approaches introduced in the previous chapter and building a baseline, here, we built a 2-layer graph convolutional network for this task. In this model, we used our new embeddings as the representations to evaluate the performance of our model in comparison with another context-based language model, BERT and SBERT. The results showed our GCN model performed well in the classification task in comparison with other models and representations. We have validated all the experiments on a 10-fold cross-validation process. We also performed an empirical performance comparison of the GCN model when using half of the dataset, and the results showed the robustness of the model. The main reason the GCN worked well on this categorization task is that this model computes the features of each node as the weighted average of itself and its second-order neighbors, so it outperformed numerous state-of-the-art methods on this dataset.

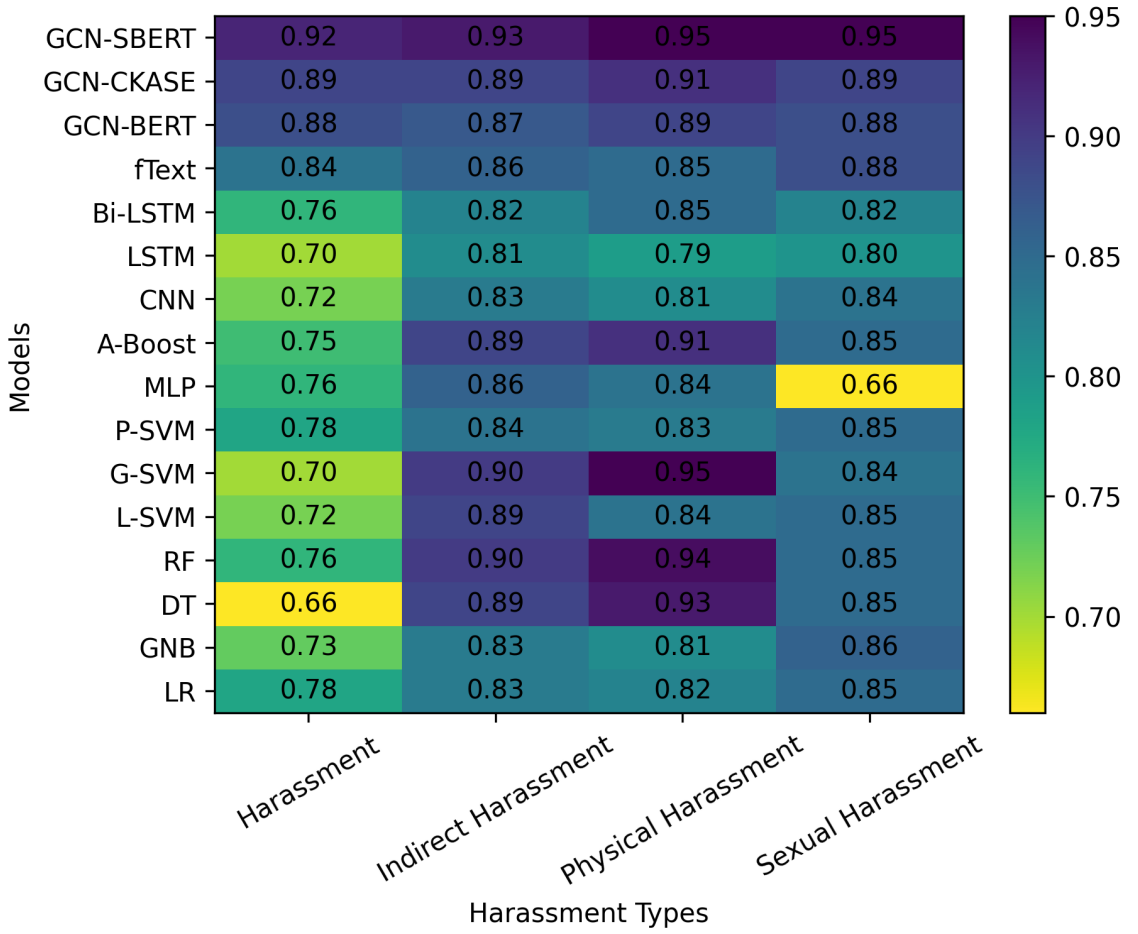


Figure 5.2: Accuracy heatmap of each one of the models (in rows) on different harassment types (in columns). The lighter color indicates lower accuracy.

Model	Harassment	Indirect Harassment	Physical Harassment	Sexual Harassment
LR	78± 0.65	83 ± 0.12	82 ± 0.02	85± 0.41
GNB	73± 0.48	83 ± 0.54	81 ± 0.37	86± 0.67
DT	66± 0.71	89 ± 0.52	93 ± 0.85	85± 0.03
RF	76± 0.55	90 ± 0.48	94 ± 0.35	85± 0.54
L-SVM	72± 0.96	89 ± 0.23	84 ± 0.24	85± 0.08
G-SVM	70± 0.52	90 ± 0.33	95 ± 0.96	84± 0.61
P-SVM	78± 0.65	84 ± 0.56	83 ± 0.71	85± 0.07
MLP	76± 0.54	86 ± 0.32	84 ± 0.76	66± 0.22
ABoost	75± 0.61	89 ± 0.19	91 ± 0.54	85± 0.54
CNN	72± 0.54	83 ± 0.45	81 ± 0.56	84± 0.76
LSTM	70± 0.05	81 ± 0.68	79 ± 0.34	80± 0.23
Bi-LSTM	76± 0.34	82 ± 0.34	85 ± 0.43	82± 0.65
fText	84± 0.17	86 ± 0.93	85 ± 0.05	88± 0.54
GCN-BERT	88± 0.21	87 ± 0.14	89 ± 0.53	88± 0.48
GCN-CKASE	89± 0.53	88 ± 0.14	91 ± 0.22	89± 0.43
GCN-SBERT	92± 0.32	93 ± 0.15	95 ± 0.28	95± 0.04

Table 5.1: Test Accuracy on both categorization tasks. We ran all models 10 times and reported the mean ± standard deviation. GCN significantly outperforms other models on this Tweet dataset, based on student t-test ($p<0.05$).

Model	Harassment	Indirect Harassment	Physical Harassment	Sexual Harassment
GCN	92± 0.32	93± 0.56	95± 0.16	95± 0.54
GCN*	90± 0.53	91± 0.76	92± 0.73	92± 0.02

Table 5.2: Test Accuracy of the model when the data is complete (GCN), in comparison with when the size of data is halved (GCN*). Here, we used SBERT embeddings which is the winner of the previous experiments.

Chapter 6

Conclusions and Future Research

This dissertation presents a novel text representation learning model considering the contextual information in the representations. We continue showing the applications of this new representation model in two NLP tasks, WSD and text classification. We build this representation learning model based on our results from analyzing recent state-of-the-art language models on parts of speech tags in the WSD task. Our representation learning model considers context from the surrounding words of the input text and the context from knowledge bases. We compare our model against baselines, including recent contextual language models. Using our representations in a 1-nearest-neighbor approach to solve the WSD task, we show that our representations significantly improve the accuracy of the final results.

Second, we apply our representations in a text classification task, classifying harassment text. We divide this task into two levels, binary and multi-class classification tasks. We show that using our representations in a deep learning model has better classification accuracy performance than other models. Finally, we build a 2-layer graph convolutional network for the multi-class classification of harassment text. This model considers the weighted average of each node's features and the second-order neighboring nodes. The experimental results show that involving neighboring information as context, in terms of considering sentence embeddings when classifying the text, helps to improve the final results.

The following contribution of this thesis is an algorithmic contribution to the Wikification task. Wikification is the WSD task when Wikipedia is the knowledge base. We present two algorithms for Wikification. The first algorithm segments the text and uses contextual information from overlapped windows. It helps to augment the contextual information of the input text when disambiguating the text. The second algorithm assigns weights to possible meanings of ambiguous words and ranks them based on the relevancy degree of each possible meaning to the context. Our results of comparison with other recent state-of-the-art models show the effectiveness of our algorithms.

There are multiple future directions to follow from this thesis, described below.

WSD problem: This thesis evaluates our approaches regarding the WSD problem to general English and domain-specific biomedical texts. In the biomedical domain, we know that the UMLS metathesaurus is disambiguating terms in this field more accurately than when we use Wikipedia [175, 213], and MetaMap is the most widely used tool in this field to disambiguate the concepts [43]. The following are some open questions and ideas:

- (i) How is the link structure between UMLS's entities that helped this source to be beneficial for the task of WSD on biomedical text? In other domains, if there is no such a knowledge base, can we build one with a similar link structure between the knowledge base's entities of domain-specific fields?

- (ii) Is it possible to combine the knowledge bases of Wikipedia and UMLS for better disambiguation accuracy compared with the MetaMap tool? Using the link structure between synsets of BabelNet through the hypernyms and hyponyms relation helped design embeddings and disambiguate the general English texts. The availability of a similar link structure in domain-specific knowledge bases might also improve the final results.

Text Disambiguation: The second challenge regards the method of text disambiguation. Our current work provided a new context-based representation learning model that produced sense representations by considering the surrounding words of each concept and applying the 1-nearest-neighbor approach for WSD. This consideration of context improved the performance in different NLP tasks, including WSD. We used the pre-trained language model BERT in our proposed model to generate the initial representations. Then, we concatenated the representation of each sense with the context representation of that sense. We used this representation in a 1-nearest-neighbor approach to disambiguate the text. One possible future research is to develop the graph convolution network during the disambiguation time, the same strategy as our model in the classification task. The objective is to disambiguate the text using our new representation as input to a deep learning model. We can build the graph of the mentions (as nodes) in the text with weighted edges. The weights are the cosine similarity of the two nodes. Using the graph convolutional network model would enhance the algorithm to learn the neighboring information as a form of context at each disambiguation step. Involving neighboring information from the graph structure as another

form of context would improve the accuracy of finding the correct meaning of the ambiguous words since we observed that considering context from the input text and knowledge base at the time of disambiguation improves disambiguation accuracy. In summary, we can solve the disambiguation task using our representations and a graph convolutional network architecture as the next step.

Domain-Specific Model Training: This study applied the new representations of biomedical concepts when solving the WSD in this domain. If our focus is on one specific domain, like the biomedical domain, we need to train the model with the domain-specific knowledge base, like UMLS. This could be the next possible research direction to use UMLS when training the new representation learning model. This idea involves bringing more contextual information from the same domain. In this scenario, we get closer to the general view of involving context in representations when disambiguating the text. Since this context type is based on the specified domain, it would increase the accuracy of the results of domain-specific text disambiguation.

In general, considering other resources like Wikidata, that have better coverage of named entities than Wordnet is an interesing future direction to continue the Wikification task.

Text Classification: Building upon the foundation laid in Chapter 4, chapter 5 investigated the potential of various models, including classical machine learning and deep neural networks, to improve classification performance on Twitter data. We introduced a two-layer GCN, specifically designed for this task, and utilized our novel embeddings to assess its effectiveness against other context-aware language models, such as BERT and SBERT. Employing a robust 10-fold cross-validation process, we demonstrated that the GCN model consistently outperformed other approaches, showcasing its superior classification capabilities. Furthermore, empirical analysis with a reduced dataset confirmed the model's inherent robustness. The GCN's remarkable performance can be attributed to its unique feature computation method, which incorporates information from both the node itself and its second-order neighbors, enabling it to surpass numerous state-of-the-art methods.

While our study demonstrates the effectiveness of our GCN model for Twitter data classification, future work will focus on pushing the boundaries of this approach. One key direction is to explore the integration of additional information beyond text into the GCN model. This could include user information, such as follower/following networks,

or external knowledge sources, such as sentiment lexicons. Additionally, we will investigate incorporating more sophisticated graph attention mechanisms to capture complex relationships between nodes.

One of the major challenges we faced was the limited availability of labeled data for Twitter harassment classification. This resulted in the need for careful data augmentation and attention to model regularization to avoid overfitting. Moving forward, we will explore active learning techniques to efficiently collect more labeled data and improve the model's generalizability.

Furthermore, we aim to extend our model to handle other types of text data and tasks beyond Twitter and harassment classification. This will involve adapting the model architecture and feature engineering to the specific characteristics of different datasets and tasks. Additionally, we will explore the potential of our GCN model for multi-label classification and other NLP tasks.

By addressing these challenges and pursuing these future directions, we believe our GCN model has the potential to become a powerful tool for various NLP applications and contribute significantly to the field of natural language processing.

In this thesis, we applied the proposed representation learning model to WSD and text classification tasks. Another possible future direction is to use this representation learning model in similar NLP tasks, including named entity recognition [190, 146], relation extraction [149], topic modeling [77], question answering [72], and sentiment and emotion analysis [205], to name a few.

Bibliography

[1] Ahmad Aghaebrahimian and Mark Cieliebak. Named entity disambiguation at scale. In *9th IAPR TC3 Workshop, ANNPR 2020, Artificial Neural Networks for Pattern Recognition (ANNPR'20), Winterthur, Switzerland, 2-4 September 2020*, pages 102–110. Springer, 2020.

[2] Eneko Agirre, Lluís Màrquez, and Richard Wicentowski. Fourth international workshop on semantic evaluations. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, Prague, Czech Republic, June 2007. ACL.

[3] Eneko Agirre, David Martínez, Oier López de Lacalle, and Aitor Soroa. Two graph-based algorithms for state-of-the-art wsd. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 585–593, 2006.

[4] Desislava Aleksandrova, Patrick Drouin, Francois Lareau, and Antoine Venant. Automatic multilingual utterance detection biased in wikipedia. *Proceedings of Recent Advances in Natural Language Processing, ACL*, page 42–51, September 2019, Bulgaria.

[5] Liz Amos, David Anderson, Stacy Brody, Anna Ripple, and Betsy L Humphreys. Umls users and uses: a current overview. *Journal of the American Medical Informatics Association*, 27(10):1606–1611, 2020.

[6] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion (WWW), Perth Canada*, pages 759–760. International World Wide Web Conferences Steering Committee, 2017.

[7] Gerard Escudero Bakx, LM Villodre, and GR Claramunt. Machine learning techniques for word sense disambiguation. *Unpublished doctoral dissertation, Universitat Politecnica de Catalunya*, Barcelona, 2006.

[8] Timo Baumann, Arne Köhn, and Felix Hennig. The spoken wikipedia corpus collection: Harvesting, alignment and an application to hyperlistening. *Language Resources and Evaluation*, 53(2):303–329, 2019.

[9] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[10] Michele Bevilacqua, Tommaso Pasini, Alessandro Raganato, Roberto Navigli, et al. Recent trends in word sense disambiguation: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. International Joint Conference on Artificial Intelligence, Montreal Canada, August 2021.

[11] Krishnanjan Bhattacharjee, S ShivaKarthik, Swati Mehta, Ajai Kumar, Snehal Phatangare, Kirti Pawar, Sneha Ukarande, Disha Wankhede, and Devika Verma. Survey and gap analysis of word sense disambiguation approaches on unstructured texts. In *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pages 323–327. IEEE, 2020.

[12] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[13] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[14] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *Artificial intelligence and statistics*, pages 127–135. Proceedings of Machine Learning Research (PMLR), 2012.

[15] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[16] Gerlof Bouma. Normalized (pointwise) mutual information in collocation extraction. *Processing Texts Automatically, GSCL Conference 2009, Tubingen*, pages 31–40, 2009.

[17] Leo Breiman. Bagging predictors. *Journal of Machine learning*, 24(2):123–140, 1996.

[18] Uwe Bretschneider, Thomas Wöhner, and Ralf Peters. Detecting online harassment in social networks. In *Proceedings of the International Conference on Information Systems - Building a Better World through Information Systems, ICIS 2014, Auckland, New Zealand, December 14-17*, 2014.

[19] Uwe Bretschneider, Thomas Wöhner, and Ralf Peters. Detecting online harassment in social networks. *In ICIS 2014 Proceedings: Conference Theme Track: Building a Better World through ICIS*, December 14-17, 2014.

[20] Susan Windisch Brown. *Finding meaning: Sense inventories for improved word sense disambiguation*. University of Colorado at Boulder, 2010.

[21] Alexander Budanitsky. Lexical semantic relatedness and its application in natural language processing. *Technical Report CSRG-390, Computer Systems Research Group, University of Toronto, ACM*, August 1999.

[22] Razvan Bunescu and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. *11st Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, April 3-7, 2006, Trento, Italy*, pages 9–16, 2006.

[23] Hiram Calvo, Arturo P Rocha-Ramírez, Marco A Moreno-Armendáriz, and Carlos A Duchanoy. Toward universal word sense disambiguation using deep neural networks. *Institute of Electrical and Electronics Engineers (IEEE) Access*, 7:60264–60275, 2019.

[24] Jose Camacho-Collados and Mohammad Taher Pilehvar. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788, 2018.

[25] José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. NASARI: a novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[26] José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64, 2016.

[27] Dhivya Chandrasekaran and Vijay Mago. Evolution of semantic similarity—a survey. *ACM Computing Surveys (CSUR)*, 54(2):1–37, 2021.

[28] Angel X Chang, Valentin I Spitkovsky, Christopher D Manning, and Eneko Agirre. Evaluating the word-expert approach for named-entity disambiguation. *Arxiv, /1603.04767*, 2016.

[29] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[30] Chi-hau Chen. Handbook of pattern recognition and computer vision. *Journal of World Scientific*, page 207–248, 2015.

[31] Yahui Chen. Convolutional neural network for sentence classification. Master's thesis, University of Waterloo, 2015.

[32] Rudi L Cilibrasi and Paul MB Vitanyi. The google similarity distance. *Institute of Electrical and Electronics Engineers (IEEE) Transactions on knowledge and data engineering*, 19(3):370–383, 2007.

[33] Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *In Proceedings of the 56th Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, page 2126–2136, 2018, Melbourne, Australia.

[34] Edilson A Correa Jr, Alneu A Lopes, and Diego R Amancio. Word sense disambiguation: A complex network approach. *Information Sciences*, 442:103–113, 2018.

[35] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716. ACL, 2007, Prague, Czech Republic.

[36] Cedric De Boom, Steven Van Canneyt, Steven Bohez, Thomas Demeester, and Bart Dhoedt. Learning semantic similarity for very short texts. In *2015 ieee international conference on data mining workshop (icdmw)*, pages 1229–1234. IEEE, 2015.

[37] Rushali Dhumal Deshmukh and Arvind Kiwelekar. Deep learning techniques for part of speech tagging by natural language processing. In *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pages 76–81. IEEE, 2020.

[38] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *In North American Association for Computational Linguistics (NAACL)*, 2018, New Orleans, Louisiana.

[39] Vimal Dixit, Kamlesh Dutta, and Pardeep Singh. Word sense disambiguation and its approaches. *CPUH-Research Journal (Career Point University Hamirpur)*, 1(2):54–58, 2015.

[40] Andres Duque, Mark Stevenson, Juan Martinez-Romo, and Lourdes Araujo. Co-occurrence graphs for word sense disambiguation in the biomedical domain. *Artificial intelligence in medicine*, 87:9–19, 2018.

[41] Philip Edmonds and Scott Cotton. Senseval-2: overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, 2001, Toulouse, France.

[42] Philip Edmonds and Scott Cotton. SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, Toulouse, France, July 2001, Toulouse, France. Association for Computational Linguistics.

[43] Jung-Wei Fan and Carol Friedman. Word sense disambiguation via semantic type classification. In *AMIA Annual Symposium Proceedings*, volume 2008, page 177. American Medical Informatics Association, 2008.

[44] Christiane Fellbaum. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer, 2010.

[45] Paolo Ferragina and Ugo Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM, 2010, Toronto Canada.

[46] Raoni S Ferreira, Maria da Graça Pimentel, and Marco Cristo. A wikification prediction model based on the combination of latent, dyadic, and monadic features. *Journal of the Association for Information Science and Technology*, 69(3):380–394, 2018.

[47] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, pages 148–156, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[48] Travis Gagie, Mozhgan Saeidi, and Allan Sapucaia. Ruler wrapping. *International Journal of Computational Geometry & Applications*, 33(01n02):3–12, 2023.

[49] William A Gale, Kenneth W Church, and David Yarowsky. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26(5):415–439, 1992.

[50] Björn Gambäck and Utpal Kumar Sikdar. Using convolutional neural networks to classify hate-speech. In *Proceedings of the first workshop on abusive language online, Association for Computational Linguistics, Vancouver, Canada*, pages 85–90, 2017.

[51] Yoav Goldberg. Assessing bert's syntactic abilities. *Computer Research Repository (CoRR), abs/1901.05287*, 2019.

[52] Thomas L Griffiths, Mark Steyvers, and Joshua B Tenenbaum. Topics in semantic representation. *Psychological review*, 114(2):211, 2007.

[53] Luis Guerra, Laura M McGarry, Víctor Robles, Concha Bielza, Pedro Larrañaga, and Rafael Yuste. Comparison between supervised and unsupervised classifications of neuronal cell types: a case study. *Developmental neurobiology*, 71(1):71–82, 2011.

[54] Wenzhong Guo, Jianwen Wang, and Shiping Wang. Deep multimodal representation learning: A survey. *IEEE Access*, 7:63373–63394, 2019.

[55] Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R Curran. Evaluating entity linking with wikipedia. *Artificial intelligence*, 194:130–150, 2013.

[56] Christian Hadiwinoto, Hwee Tou Ng, and Wee Chung Gan. Improved word sense disambiguation using pre-trained contextualized word representations. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) Hong Kong*, page 5300–5309, 2019.

[57] El Hannach Hajar and Benkhalifa Mohammed. Using synonym and definition wordnet semantic relations for implicit aspect identification in sentiment analysis. In *Proceedings of the 2nd International Conference on Networking, Information Systems & Security, Morocco*, pages 1–5. Association for Computing Machinery, 2019.

[58] N Sree Harsha, Ch Nageswar Kumar, Vijaya Krishna Sonthi, and K Amarendra. Lexical ambiguity in natural language processing applications. In *2022 International Conference on Electronics and Renewable Systems (ICEARS)*, pages 1550–1555. Institute of Electrical and Electronics Engineers (IEEE), Thoothukudi, Tamil Nadu, India, 2022.

[59] Bradley Hauer, Grzegorz Kondrak, Yixing Luan, Arnob Mallik, and Lili Mou. Semi-supervised and unsupervised sense annotation via translations. *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 504–513, 2021.

[60] Haibo He and Yunqian Ma. Book: Imbalanced learning: foundations, algorithms, and applications. *Wiley-IEEE Press*, 2013.

[61] Angelos Hliaoutakis, Giannis Varelas, Epimenidis Voutsakis, Euripides GM Petrakis, and Evangelos Milios. Information retrieval by semantic similarity. *International journal on semantic Web and information systems (IJSWIS)*, 2(3):55–73, 2006.

[62] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Journal of Neural computation*, 9(8):1735–1780, 1997.

[63] Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. Kore: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 110–119, New Orleans, USA, 2012.

[64] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, UK*, pages 782–792. ACL, 2011.

[65] Cheng-Hui Huang, Jian Yin, and Fang Hou. A text similarity measurement combining word semantic information with tf-idf method. *Jisuanji Xuebao(Chinese Journal of Computers)*, 34(5):856–864, 2011.

[66] Qi Huang, Chuan Zhou, Jia Wu, Mingwen Wang, and Bin Wang. Deep structure learning for rumor detection on twitter. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[67] Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Germany, pages 897–907, 2016.

[68] Sho Ishida, Tomo Miyazaki, Yoshihiro Sugaya, and Shinichiro Omachi. Graph neural networks with multiple feature extraction paths for chemical property estimation. *Molecules*, 26(11):3125, 2021.

[69] Tobias Skovgaard Jepsen. How to do deep learning on graphs with graph convolutional networks. *Towards Data Science Webpage*, 18, September 2018.

[70] Akshita Jha and Radhika Mamidi. When does a compliment become sexist? analysis and classification of ambivalent sexism using twitter data. In *Proceedings of the second workshop on NLP and computational social science*, pages 7–16, Vancouver, Canada, 2017.

[71] Antonio J Jimeno-Yepes, Bridget T McInnes, and Alan R Aronson. Exploiting mesh indexing in medline to generate a data set for word sense disambiguation. *BMC bioinformatics*, 12(1):1–14, 2011.

[72] Qiao Jin, Zheng Yuan, Guangzhi Xiong, Qianlan Yu, Huaiyuan Ying, Chuanqi Tan, Mosha Chen, Songfang Huang, Xiaozhong Liu, and Sheng Yu. Biomedical question answering: A survey of approaches and challenges. *Journal of ACM Computing Surveys (CSUR)*, 55(2):1–36, 2022.

[73] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, pages 53–60, 1972.

[74] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *Transactions of the Association for Computational Linguistics (TACL)*, 53:135–146, 2016.

[75] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *ArXiv, https://arxiv.org/abs/1602.02410*, 2016.

[76] Pooja Kamavisdar, Sonam Saluja, and Sonu Agrawal. A survey on image classification approaches and techniques. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(1):1005–1009, 2013.

[77] Pooja Kherwa and Poonam Bansal. Topic modeling: a comprehensive review. *Journal of EAI Endorsed transactions on scalable information systems*, 7(24), 2019.

[78] Meen Chul Kim, Seojin Nam, Fei Wang, and Yongjun Zhu. Mapping scientific landscapes in umls research: a scientometric review. *Journal of the American Medical Informatics Association*, 27(10):1612–1624, 2020.

[79] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[80] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *In International Conference on Learning Representations (ICLR), CoRR, abs/1412.6980, Banff Canada*, 2014.

[81] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR), San Juan, Puerto Rico*, pages 1–14, 2016.

[82] Dijana Kosmajac, Stacey Taylor, and Mozhgan Saeidi. Dnlp@ fintoc'20: Table of contents detection in financial documents. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 169–173, 2020.

[83] Zeljko Kraljevic, Daniel Bean, Aurelie Mascio, Lukasz Roguski, Amos Folarin, Angus Roberts, Rebecca Bendayan, and Richard Dobson. Medcat–medical concept annotation tool. *arXiv preprint arXiv:1912.10166*, 2019.

[84] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–466. ACM, Paris France, 2009.

[85] S Kumar, M Anand Kumar, and KP Soman. Deep learning based part-of-speech tagging for malayalam twitter data (special issue: deep learning techniques for natural language processing). *Journal of Intelligent Systems*, 28(3):423–435, 2019.

[86] Sawan Kumar, Sharmistha Jat, Karan Saxena, and Partha Talukdar. Zero-shot word sense disambiguation using sense definition embeddings. In *Proceedings of the 57th Annual Meeting of the ACL, Italy*, pages 5670–5681, 2019.

[87] Sunjae Kwon, Dongsuk Oh, and Youngjoong Ko. Word sense disambiguation based on context selection using knowledge-based word similarity. *Journal of Information Processing & Management*, 58(4):102551, 2021.

[88] Pierre Lafon. Sur la variabilité de la fréquence des formes dans un corpus. *Mots. Les langages du politique*, 1(1):127–165, 1980.

[89] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[90] Pedro Larranaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Inaki Inza, José A Lozano, Rubén Armananzas, Guzmán Santafé, Aritz Pérez, et al. Machine learning in bioinformatics. *Briefings in bioinformatics*, 7(1):86–112, 2006.

[91] Nevena Lazic, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. Plato: A selective context model for entity resolution. *Transactions of the Association for Computational Linguistics, Vietnam*, 3:503–515, 2015.

[92] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[93] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[94] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM, New York US, 1986.

[95] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems*, 27:2177–2185, 2014.

[96] Bing Li. Named entity recognition in the style of object detection. *arXiv preprint arXiv:2101.11122*, 2021.

[97] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*, volume 32, pages 234–242, New Orleans, 2018. Proceedings of the AAAI Conference on Artificial Intelligence.

[98] Hong Liang, Xiao Sun, Yunlei Sun, and Yuan Gao. Text feature extraction based on deep learning: a review. *EURASIP journal on wireless communications and networking*, 2017(1):1–12, 2017.

[99] Geng Lin and Jian Guan. Solving maximum set k-covering problem by an adaptive binary particle swarm optimization method. *Knowledge-Based Systems*, 142:95–107, 2018.

[100] Marek Lipczak, Arash Koushkestani, and Evangelos Milios. Tulip: lightweight entity recognition and disambiguation using wikipedia-based topic centroids. In *Proceedings of the first international workshop on Entity recognition & disambiguation, SIGIR '14: The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval Gold Coast Queensland Australia*, pages 31–36, 2014.

[101] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. *In Proceedings of the TwentyFifth International Joint Conference on Artificial Intelligence, IJCAI'16, New York, US*, page 2873–2879, 2016.

[102] Elena Lloret. Text summarization: an overview. *Universidad de Alicante Alicante, Spain, Paper supported by the Spanish Government under the project TEXT-MESS (TIN2006-15265-C06-01)*, 2008.

[103] Daniel Loureiro and Jose Camacho-Collados. Don't neglect the obvious: on the role of unambiguous words in word sense disambiguation. *arXiv preprint arXiv:2004.14325*, 2020.

[104] Daniel Loureiro and Alipio Jorge. Language modelling makes sense: Propagating representations through wordnet for full-coverage word sense disambiguation. *In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.Florence, Italy.*, page 5682–5691, 2019.

[105] Daniel Loureiro and Alípio Mário Jorge. Liaad at semdeep-5 challenge: Word-in-context (wic). In *SemDeep@IJCAI*, pages 31–39, china, 2019. SemDeep@IJCAI.

[106] Daniel Loureiro, Kiamehr Rezaee, Mohammad Taher Pilehvar, and Jose Camacho-Collados. Analysis and evaluation of language models for word sense disambiguation. *Computational Linguistics*, 47(2):387–443, 2021.

[107] Xinghua Lu, Bin Zheng, Atulya Velivelli, and ChengXiang Zhai. Enhancing text categorization with semantic-enriched representation and training data augmentation. *Journal of the American Medical Informatics Association*, 13(5):526–535, 2006.

[108] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge university press, 2008.

[109] Yuqing Mao and Kin Wah Fung. Use of word and graph embedding to measure semantic relatedness between unified medical language system concepts. *Journal of the American Medical Informatics Association*, 27(10):1538–1546, 2020.

[110] Tamara Martín Wanton and Rafael Berlanga Llavori. A clustering-based approach for unsupervised word sense disambiguation. *Espola para*, 2012.

[111] Angel R Martinez. Part-of-speech tagging. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(1):107–113, 2012.

[112] Jose L Martinez-Rodriguez, Aidan Hogan, and Ivan Lopez-Arevalo. Information extraction meets the semantic web: a survey. *Semantic Web*, Preprint:1–81, 2020.

[113] Diana McCarthy, Rob Koeling, Julie Weeds, and John A Carroll. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 279–286, 2004.

[114] Sara Meftah and Nasredine Semmar. A neural network model for part-of-speech tagging of social media texts. In *Proceedings of the eleventh international Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[115] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning, Berlin, Germany*, pages 51–61, 2016.

[116] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems, Graz Austria*, pages 1–8, 2011.

[117] Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780, 2006.

[118] Rada Mihalcea and Andras Csomai. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, Lisbon Portugal*, pages 233–242. ACM, 2007.

[119] Andrei Mikheev, Marc Moens, and Claire Grover. Named entity recognition without gazetteers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics, Norway*, pages 331–339. ACL, 1999.

[120] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Proceedings of International Conference on Learning Representations, ICLR, Arizona*, 2013.

[121] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

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

[123] George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244, 1990.

[124] George A Miller, Claudia Leacock, Randee Tengi, and Ross T Bunker. A semantic concordance. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*, 1993.

[125] David Milne and Ian H Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management, California*, pages 509–518, 2008.

[126] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning–based text classification: a comprehensive review. *ACM computing surveys (CSUR)*, 54(3):1–40, 2021.

[127] Zhao-Yan Ming and Tat Seng Chua. Resolving polysemy and pseudonymity in entity linking with comprehensive name and context modeling. *Information Sciences*, 307:18–38, 2015.

[128] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[129] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. *Advances in neural information processing systems*, 26:2265–2273, 2013.

[130] Lorenza Moreno-Monteagudo, Rubén Izquierdo-Beviá, Patricio Martínez-Barco, and Armando Suárez. A study of the influence of pos tagging on wsd. In *Text, Speech and Dialogue: 9th International Conference, TSD 2006, Brno, Czech Republic, September 11-15, 2006. Proceedings 9*, pages 173–179. Springer, 2006.

[131] Andrea Moro and Roberto Navigli. SemEval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), ACL, Colorado*, pages 288–297, Denver, Colorado, June 2015. Association for Computational Linguistics.

[132] Munirsyah Munirsyah, Moch Arif Bijaksana, and Widi Astuti. Development synonym set for the english wordnet using the method of comutative and agglomerative clustering. *Jurnal Sisfokom (Sistem Informasi dan Komputer)*, 9(2):171–176, 2020.

[133] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[134] Kotaro Nakayama, Takahiro Hara, and Shojiro Nishio. Wikipedia link structure and text mining for semantic relation extraction. In *SemSearch*, pages 59–73. ACL, 2008.

[135] Fatemeh Nargesian, Horst Samulowitz, Udayan Khurana, Elias B Khalil, and Deepak S Turaga. Learning feature engineering for classification. In *Ijcai*, volume 17, pages 2529–2535, 2017.

[136] Usman Naseem, Imran Razzak, Shah Khalid Khan, and Mukesh Prasad. A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *Transactions on Asian and Low-Resource Language Information Processing*, 20(5):1–35, 2021.

[137] Roberto Navigli. Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 41(2):1–69, 2009.

[138] Roberto Navigli, David Jurgens, and Daniele Vannella. SemEval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics.

[139] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial intelligence*, 193:217–250, 2012.

[140] Roberto Navigli and Daniele Vannella. Semeval-2013 task 11: Word sense induction and disambiguation within an end-user application. In *Second Joint Conference on Lexical and Computational Semantics (* SEM) Georgia, Atlanta, Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 193–201, 2013.

[141] Arjun Srinivas Nayak, Ananthu P Kanive, Naveen Chandavekar, and R Balasubramani. Survey on pre-processing techniques for text mining. *International Journal of Engineering and Computer Science*, 5(6):16875–16879, 2016.

[142] Andrew Y Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78, 2004.

[143] Dat Ba Nguyen, Johannes Hoffart, Martin Theobald, and Gerhard Weikum. Aida-light: High-throughput named-entity disambiguation. *Linked Data On the Web (LDOW), Korea*, 14:22–32, 2014.

[144] Hien T Nguyen and Tru H Cao. Named entity disambiguation: A hybrid approach. *International Journal of Computational Intelligence Systems*, 5(6):1052–1067, 2012.

[145] Houda Oufaida, Philippe Blache, and Omar Nouali. A coherence model for sentence ordering. In *International Conference on Applications of Natural Language to Information Systems*, pages 261–273. NLPIS, Germany, 2019.

[146] Naveen S Pagad and N Pradeep. Clinical named entity recognition methods: an overview. In *International Conference on Innovative Computing and Communications, India*, pages 151–165. Springer, 2022.

[147] Mrutyunjaya Panda. Developing an efficient text pre-processing method with sparse generative naive bayes for text mining. *International Journal of Modern Education and Computer Science*, 11(9):11, 2018.

[148] Babita Pandey, Devendra Kumar Pandey, Brijendra Pratap Mishra, and Wasiur Rhmann. A comprehensive survey of deep learning in the field of medical imaging and medical natural language processing: Challenges and research directions. *Journal of King Saud University-Computer and Information Sciences*, 2021.

[149] Sachin Pawar, Girish K Palshikar, and Pushpak Bhattacharyya. Relation extraction: A survey. *arXiv preprint arXiv:1712.05191*, 2017.

[150] Ted Pedersen. Unsupervised corpus-based methods for wsd. *Word sense disambiguation: Algorithms and applications*, pages 133–166, 2006.

[151] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), Doha, Qatar*, pages 1532–1543, 2014.

[152] Ahmad Pesaranghader, Stan Matwin, Marina Sokolova, and Ali Pesaranghader. deepbiowsd: effective deep neural word sense disambiguation of biomedical text data. *Journal of the American Medical Informatics Association*, 26(5):438–446, 2019.

[153] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *Association for Computational Linguistics*, pages 2227–2237, 2018.

[154] Francesco Piccinno and Paolo Ferragina. From tagme to wat: a new entity annotator. In *Proceedings of the first international workshop on Entity recognition & disambiguation, SIGIR '14: The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval Gold Coast Queensland Australia*, pages 55–62. ACM, 2014.

[155] Mohammad Taher Pilehvar, Jose Camacho-Collados, Roberto Navigli, and Nigel Collier. Towards a seamless integration of word senses into downstream nlp applications. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL, Vancouver, Canada*, 2017.

[156] Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Sofia, Bulgaria*, pages 1341–1351, 2013.

[157] Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. SemEval-2007 task-17: English lexical sample, SRL and all words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 87–92, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[158] Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP), Copenhagen, Denmark*, pages 1156–1167, 2017.

[159] Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. Automatic construction and evaluation of a large semantically enriched wikipedia. In *25th International Joint Conference on Artificial Intelligence IJCAI-16!*, pages 2894–2900. IJCAI, New York City, 2016.

[160] Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, EACL, Valencia, Spain*, pages 99–110, 2017.

[161] Nazreena Rahman and Bhogeswar Borah. An unsupervised method for word sense disambiguation. *Journal of King Saud University-Computer and Information Sciences*, 34(9):6643–6651, 2022.

[162] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, Portland Oregon*, pages 1375–1384. Association for Computational Linguistics, 2011.

[163] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), Hong Kong*, 2019.

[164] Henry Rosales-Méndez, Bárbara Poblete Labra, and Aidan Hogan. What should entity linking link? In *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, May 21-25, 2018*, pages 55–62. CEUR-WS, Cali, Colombia, 2018.

[165] AKM Sabbir, Antonio Jimeno-Yepes, and Ramakanth Kavuluru. Knowledge-based biomedical word sense disambiguation with neural concept embeddings. In *2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 163–170. IEEE, 2017.

[166] Mozhgan Saeidi. Contextbert: Contextual graph representation learning in text disambiguation. *Machine Learning with Symbolic Methods and Knowledge Graphs, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2021), Pages 15-29, September 13-17, 2021, Spain*, 2021.

[167] Mozhgan Saeidi, Kaveh Mahdaviani, Evangelos Milios, and Norbert Zeh. Context-enhanced concept disambiguation in wikification. *Intelligent Systems with Applications*, pages 200–246, 2023.

[168] Mozhgan Saeidi, Evangelos Milios, and Norbert Zeh. Contextualized knowledge base sense embeddings in word sense disambiguation. In *Workshop on Machine Learning (WML 2021, 3rd edition) 2021, 16th International Conference on Document Analysis and Recognition (ICDAR), 5-10 September 2021, Lausanne, Switzerland*, pages 174–186. Springer, 2021.

[169] Mozhgan Saeidi, Evangelos Milios, and Norbert Zeh. Graph convolutional networks for categorizing online harassment on twitter. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 946–951. IEEE, 2021.

[170] Mozhgan Saeidi, Evangelos Milios, and Norbert Zeh. Graph representation learning in document wikification. In *Document Analysis and Recognition–ICDAR 2021 Workshops: Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II 16*, pages 509–524. Springer, 2021.

[171] Mozhgan Saeidi, Evangelos Milios, and Norbert Zeh. Biomedical word sense disambiguation with contextualized representation learning. In *Companion Proceedings of the Web Conference 2022*, pages 843–848, 2022.

[172] Mozhgan Saeidi, Samuel Bruno da S Sousa, Evangelos Milios, Norbert Zeh, and Lilian Berton. Categorizing online harassment on twitter. In *SocIaL Media And Harassment (SIMAH), Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2019), Communications in Computer and Information Science, vol 1168, September 2019, Germany,*, pages 283–297. springer, Germany, 2019.

[173] Armin Sajadi. *Semantic Analysis using Wikipedia Graph Structure*. PhD thesis, Dalhousie University, 2018.

[174] Armin Sajadi, Evangelos E Milios, and Vlado Keselj. Vector space representation of concepts using wikipedia graph structure. In *International Conference on Applications of Natural Language to Information Systems*, pages 393–405. Springer, Belgium, 2017.

[175] Armin Sajadi, Evangelos E Milios, Vlado Kešelj, and Jeannette CM Janssen. Domain-specific semantic relatedness from wikipedia structure: A case study in biomedical text. In *International conference on intelligent text processing and computational linguistics*, pages 347–360. Springer, 2015.

[176] Gerard Salton. Developments in automatic text retrieval. *Journal of Science*, 253(5023):974–980, 1991.

[177] Gerard Salton and Michael J McGill. *Introduction to modern information retrieval*. McGraw-Hill, Inc., 1986.

[178] Gerard Salton and Chung-Shu Yang. On the specification of term values in automatic indexing. *Journal of Documentation*, 1973.

[179] Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *Journal of Computer Science*, 2(3):1–21, 2021.

[180] Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. Sensembert: Context-enhanced sense embeddings for multilingual word sense disambiguation. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, volume 34, pages 8758–8765, New York, 2020.

[181] Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. With more contexts comes better performance: Contextualized sense embeddings for all-round word sense disambiguation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Virtual event*, pages 3528–3539, 2020.

[182] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[183] Craig W Schmidt. Improving a tf-idf weighted document vector embedding. *arXiv preprint arXiv:1902.09875*, 2019.

[184] Robin M Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv preprint arXiv:1912.05911*, 2019.

[185] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[186] Sima Sharifirad. Nlp and machine learning techniques to detect online harassment on social networking platforms. *PhD thesis, Dalhousie University*, 2019.

[187] Sima Sharifirad, Borna Jafarpour, and Stan Matwin. Boosting text classification performance on sexist tweets by text augmentation and text generation using a combination of knowledge graphs. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 107–114, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[188] Sima Sharifirad, Borna Jafarpour, and Stan Matwin. How is your mood when writing sexist tweets? detecting the emotion type and intensity of emotion using natural language processing techniques. *arXiv preprint arXiv:1902.03089*, 2019.

[189] Sima Sharifirad and Stan Matwin. When a tweet is actually sexist. a more comprehensive classification of different online harassment categories and the challenges in nlp. *Computing Research Repository (CoRR) abs/1902.10584*, 2019.

[190] Abhishek Sharma, Sudeshna Chakraborty, Shivam Kumar, et al. Named entity recognition in natural language processing: A systematic review. In *Proceedings of Second Doctoral Symposium on Computational Intelligence*, pages 817–828. Springer, 2022.

[191] Ilya Shnayderman, Liat Ein-Dor, Yosi Mass, Alon Halfon, Benjamin Sznajder, Artem Spector, Yoav Katz, Dafna Sheinwald, Ranit Aharonov, and Noam Slonim. Fast end-to-end wikification. *arXiv preprint arXiv:1908.06785*, 2019.

[192] Harsimran Singh and Vishal Gupta. An insight into word sense disambiguation techniques. In *International Journal of Computer Applications*, volume 118, pages 32–39. Foundation of Computer Science, 2015.

[193] Himanshu Singh and Pushpak Bhattacharyya. A survey on word sense disambiguation. *ACM computing surveys (CSUR)*, Jun 2019.

[194] Henrique Siqueira and Flavia Barros. A feature extraction process for sentiment analysis of opinions on services. In *Proceedings of International Workshop on Web and Text Intelligence*, pages 404–413, 2010.

[195] Kent A Smith. Free medline access worldwide. *Information Services & Use*, 32(Preprint):1–10, 2022.

[196] Benjamin Snyder and Martha Palmer. The english all-words task. In *Proceedings of SENSEVAL-3, The Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43. Association for Computational Linguistics, Spain, 2004.

[197] Benjamin Snyder and Martha Palmer. The English all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[198] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.

[199] Diane M Strong, Yang W Lee, and Richard Y Wang. Data quality in context. *Communications of the ACM*, 40(5):103–110, 1997.

[200] Rhea Sukthanker, Soujanya Poria, Erik Cambria, and Ramkumar Thirunavukarasu. Anaphora and coreference resolution: A review. *Information Fusion*, 59:139–162, 2020.

[201] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *In Advances in Neural Information Processing Systems (NIPS 2014)*, 2014.

[202] Andrey Sysoev and Irina Nikishina. Smart context generation for disambiguation to wikipedia. In *Conference on Artificial Intelligence and Natural Language*, pages 11–22. springer, Saint Petersburg, Russia, 2018.

[203] Julian Szymański and Maciej Naruszewicz. Review on wikification methods. *AI Communications*, 27(2):97–111, 2019.

[204] Ihsan Ullah, Mario Manzo, Mitul Shah, and Michael G Madden. Graph convolutional networks: analysis, improvements and results. *Applied Intelligence*, pages 1–12, 2021.

[205] Hande Aka Uymaz and Senem Kumova Metin. Vector based sentiment and emotion analysis from text: A survey. *Journal of Engineering Applications of Artificial Intelligence*, 113:104922, 2022.

[206] Lauren Vandenbossche, Bram Spruyt, and Gil Keppens. Young, innocent and sexist? social differences in benevolent and hostile sexist attitudes towards women amongst flemish adolescents. *Journal of YOUNG*, 26(1):51–69, 2018.

[207] Agustín Vicente and Ingrid L Falkum. Polysemy. In *Oxford research encyclopedia of linguistics*. Oxford Linguistics, 2017.

[208] Jorge Vivaldi and Horacio Rodríguez. Medical entities tagging using distant learning. In *International Conference on Intelligent Text Processing and Computational Linguistics, Budapest Hungry*, pages 631–642. springer, 2015.

[209] Jan Philip Wahle, Terry Ruas, Norman Meuschke, and Bela Gipp. Incorporating word sense disambiguation in neural language models. *arXiv preprint arXiv:2106.07967*, 2021.

[210] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3266–3280. Curran Associates, Inc., 2019.

[211] Congcong Wang, Paul Nulty, and David Lillis. A comparative study on word embeddings in deep learning for text classification. In *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval, Korea*, pages 37–46, 2020.

[212] Yinglin Wang, Ming Wang, and Hamido Fujita. Word sense disambiguation: A comprehensive knowledge exploitation framework. *Knowledge-Based Systems*, pages 105–117, 2019.

[213] Yipei Wang, Xingyu Fan, Luoxin Chen, Eric I Chang, Sophia Ananiadou, Junichi Tsujii, Yan Xu, et al. Mapping anatomical related entities to human body parts based on wikipedia in discharge summaries. *Journal of Bioinformatics and Computational Biology(BMC) bioinformatics*, 20(1):1–11, 2019.

[214] Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. Sentence similarity learning by lexical decomposition and composition. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan*, pages 1340–1349, 2016.

[215] Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop, Association for Computational Linguistics*, pages 88–93, San Diego, California, 2016.

[216] Gerhard Weikum, Luna Dong, Simon Razniewski, and Fabian Suchanek. Machine knowledge: Creation and curation of comprehensive knowledge bases. *Found. Trends Databases, 10(2-4):108–490*, 2020.

[217] Robert West, Ashwin Paranjape, and Jure Leskovec. Mining missing hyperlinks from human navigation traces: A case study of wikipedia. In *Proceedings of the 24th international conference on World Wide Web (WWW), Florence, Italy*, pages 1242–1252. ACM, 2015.

[218] Chris White. Atlantic geoscience society abstracts: 44th annual colloquium & general meeting 2018. *Atlantic Geology: Journal of the Atlantic Geoscience Society/Atlantic Geology: Revue de la Société Géoscientifique de l'Atlantique*, 54:81–132, 2018.

[219] John Wieting, Taylor Berg-Kirkpatrick, Kevin Gimpel, and Graham Neubig. Beyond bleu: training neural machine translation with semantic similarity. *arXiv preprint arXiv:1909.06694*, 2019.

[220] Kexuan Xin, Wen Hua, Yu Liu, and Xiaofang Zhou. Log: a locally-global model for entity disambiguation. *Journal of World Wide Web, Vol 24*, pages 351–373, 2020.

[221] JinXiong Yang, Liang Bai, and Yanming Guo. A survey of text classification models. In *Proceedings of the 2020 2nd International Conference on Robotics, Intelligent Control and Artificial Intelligence*, pages 327–334, 2020.

[222] Yi Yang, Wen-tau Yih, and Christopher Meek. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical mMethods in Natural Language Processing (EMNLP), Lisbon, Portugal*, pages 2013–2018, 2015.

[223] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver Canada*, 32:221–229, 2019.

[224] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, volume 33, pages 7370–7377, Honolulu, 2019. AAAI.

[225] Jiajun Zhang, Chengqing Zong, et al. Deep neural networks in machine translation: An overview. *Jouenal of IEEE Intelligent Systems*, 30(5):16–25, 2015.

[226] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.

[227] Yi Zhang, Zachary Ives, and Dan Roth. "who said it, and why?" provenance for natural language claims. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4416–4426. ACL, Virtual, 2020.

[228] Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. Semantics-aware bert for language understanding. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, volume 34, pages 9628–9635, 2020.

[229] Gang Zhao, Ji Wu, Dingding Wang, and Tao Li. Entity disambiguation to wikipedia using collective ranking. *Journal of Information Processing & Management*, 52(6):1247–1257, 2016.

[230] Arkaitz Zubiaga. Enhancing navigation on wikipedia with social tags. *arXiv preprint arXiv:1202.5469*, 2012.

Appendix A

Environment Requirements for Reproducing Results and Implementations

This part describes the implementation requirements for text disambiguation and text classification tasks.

A.1 Text Disambiguation

We use different tools and technologies to build the whole system, similar to our baseline, the Wikisim project. All the code is written in Python, presented as Jupyter Notebooks, and third-party libraries which are mentioned in the following.

- Prepare the environment: One required environment is conda. To satisfy this requirement, follow these steps to prepare the environment:

  - Install conda

  - run: `conda env create -f environment.yml`

- Clone the source code, which is written in Python. Our repository contains multiple files, but the main ones including the source code are wikisim, wikify, and vsmcoherence notebooks.

- The following requirements are MariaDB and Apache Solr servers. To prepare the MariaDB and Apache Solr servers:

  - Download MariaDB

  - Download Solr

  As it is mentioned in the Wikisim baseline [173], we also can start from scratch and import a different version of Wikipedia. It requires downloading and pre-processing the Wikipedia dumps and extracting the graph structure and textual information. The whole process can be done in two steps:

- Setting up a MariaDB server and preparing the graph structure.

The full instruction is given here: https://github.com/mozhgans/wikification/blob/master/preparation_scripts/db/prepare_graph_db.ipynb.

- Processing the text and setting up the Apache Solr.

The full instruction is given here: https://github.com/mozhgans/wikification/blob/master/preparation_scripts/text/prepare_annonated_indexed_wiki.ipynb.

- **Libraries and Dependencies**: Pandas, MySQLdb, CPickle, Scipy.

All the code related to the WSD problem and the Wikification problem studied in this thesis is available on the following links:

The GitHub repository of the code for chapter 2: https://github.com/mozhgans/wikification.

The GitHub repository of the code for chapter 3: https://github.com/mozhgans/Error-analysis-of-concept-embedding-Approaches.

A.2 Text Classification

Text Cleaning and Pre-processing: Text feature extraction and pre-processing steps for classification algorithms are crucial steps. In Natural Language Processing, most of the text and documents contain many redundant words for text classification, such as stopwords, miss-spellings, and slang. In many algorithms like statistical and probabilistic learning methods, noise and unnecessary features can negatively affect the overall performance. So, the elimination of these features is significant. This section briefly explains some techniques and methods for text cleaning and pre-processing text documents.

Tokenization

Tokenization is breaking down a text stream into words, phrases, symbols, or other meaningful elements called tokens. The main goal of this step is to extract individual words in a sentence. Therefore, it is necessary to incorporate a parser in the pipeline that performs the tokenization of the documents in text mining along with text classification. You need to run this line in order to tokenize the text:

```
from nltk.tokenize import word_tokenize
```

Stop words

Text and document classification over social media, such as Twitter, is usually affected by the text corpuses' noisy nature (abbreviations, irregular forms). You need this line to remove stop words:

```
from nltk.corpus import stopwords
```

Capitalization

Sentences can contain a mixture of uppercase and lowercase letters. Multiple sentences make up a text document. The most common approach to reducing the problem space is to reduce everything to lowercase. This brings all words in a document to the same space, but it often changes the meaning of some words, such as "US" to "us" where the first one represents the United States of America and the second one is a pronoun. To solve this, slang and abbreviation converters can be applied.

Use "text.lower" where text is your input text.

Stemming

Text Stemming is modifying a word to obtain its variants using different linguistic processes like affixation (addition of affixes). For example, the stem of the word "studying" is "study".

```
from nltk.stem import PorterStemmer
```

Lemmatization

Text lemmatization eliminates the redundant prefix or suffix of a word and extracts the base word (lemma).

```
from nltk.stem import WordNetLemmatizer
```

Libraries

- Torch, torchtext, torchvision
- Numpy
- Scipy
- Pandas

All the code related to the text classification problem studied in this thesis is available on the following links:

The GitHub repository of the code for chapter 4: https://github.com/mozhgans/Competition-Categorizing-Online-Harassment-on-Twitter.

The GitHub repository of the code for chapter 5: https://github.com/mozhgans/Classification_harassment.

Appendix B

Brief information on BabelNet

BabelNet [1] is a multilingual lexicalized semantic network and ontology that was developed by researchers at the Sapienza University of Rome and the Italian National Research Council [139]. It combines data from various lexical resources and knowledge bases, such as WordNet, Wikipedia, and Wiktionary, to create a comprehensive and interconnected network of words and concepts in multiple languages.

The main goal of BabelNet is to provide a unified and consistent representation of the meanings of words across different languages. It links together words with similar meanings and establishes relationships between them, allowing for cross-lingual information retrieval, machine translation, and other natural language processing tasks.

BabelNet assigns each word or concept a unique identifier called a "Babel synset." A Babel synset represents a group of synonymous words or phrases across multiple languages that share the same meaning. These synsets are connected through a network of semantic relations, such as hypernyms (more general concepts), hyponyms (more specific concepts), meronyms (part-whole relationships), and others.

BabelNet is widely used in various applications, including machine translation, information retrieval, question-answering systems, text classification, and semantic similarity analysis. It provides a valuable resource for researchers and developers working with multilingual natural language processing tasks, allowing them to access a rich semantic network that spans multiple languages.

An example of Babelnet:

In BabelNet, the word “car” corresponds to a Babel synset with the identifier “bn:00020660n”. This synset represents the concept of a motor vehicle designed for carrying passengers. It encompasses the meaning of “car” in various languages and is connected to other related

[1] https://babelnet.org

concepts through semantic relations.

For example, within BabelNet, the synset “bn:00020660n” for “car” is linked to several other synsets through relationships. One of the hypernyms (more general concepts) of “car” is the synset “bn:00026757n” representing “motor vehicle”. On the other hand, one of the hyponyms (more specific concepts) of “car” is the synset “bn:00051247n” representing “sports car”.

By leveraging these semantic connections in BabelNet, natural language processing systems can gain a deeper understanding of the word “car” and its relationships with other concepts in various languages.

Appendix C

NASARI Vectors and Weighted Overlap Measure

In this section, we provide a detailed example of the NASARI vector and the concept of weighted overlap measure [156] and this example python implementations. We provided the formulation of this concept in Section 2.2.2. Here, we review this concept with an example. The example starts by creating a NASARI vector for the concept of "cat". In the first step, we collect all the lexical embeddings. It means we need to create a vector of word embeddings for the words that are related to the concept of "cat". These words could include "animal", "mammal", "feline", "domestic", and "pet". We can use a pre-trained word embedding model, Word2Vec, to create these word embeddings.

Once we have the word embeddings for the related words, we go to the next step, the semantic similarity step. In this step, we compute a similarity score between each word and the concept of "cat". This can be done using a variety of methods, such as cosine similarity or Jaccard similarity. The dimensions of the NASARI vector represent the semantic properties of the concept of "cat". The following code script is an example of implementation on how to calculate the weighted overlap score between two NASARI vectors.

```
[ ]  import numpy as np

     # Create two NASARI vectors
     v1 = np.array([0.5, 0.7, 0.9, 0.8, 0.6])
     v2 = np.array([0.6, 0.8, 0.7, 0.9, 0.5])

     # Calculate the cosine similarity between the two vectors
     cosine_similarity = np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))

     # Calculate the weighted overlap score
     weighted_overlap = np.sqrt(
         np.sum(
             (rank(d, v1) + rank(d, v2)) ** -1
             for d in range(len(v1))
         )
         / np.sum((2 * i) ** -1 for i in range(len(v1)))
     )

     print(weighted_overlap)
```

Figure C.1: Python code snippets on how to calculate the weighted overlap score between two NASARI vectors.

Considering the example above, here is the code implementation details of how to calculate the weighted overlap score between two NASARI vectors, using the set of dimensions O = [“animal”, “mammal”, “feline”, “domestic”, “pet”].

```
import numpy as np

# Create two NASARI vectors
v1 = np.array([0.5, 0.7, 0.9, 0.8, 0.6])
v2 = np.array([0.6, 0.8, 0.7, 0.9, 0.5])

#Calculate the cosine similarity between two vectors for each dimension in O
cosine_similarity_animal = np.dot(v1[:, 0], v2[:, 0]) /
 (np.linalg.norm(v1[:, 0]) * np.linalg.norm(v2[:, 0]))
cosine_similarity_mammal = np.dot(v1[:, 1], v2[:, 1]) /
 (np.linalg.norm(v1[:, 1]) * np.linalg.norm(v2[:, 1]))
cosine_similarity_feline = np.dot(v1[:, 2], v2[:, 2]) /
 (np.linalg.norm(v1[:, 2]) * np.linalg.norm(v2[:, 2]))
cosine_similarity_domestic = np.dot(v1[:, 3], v2[:, 3])
/ (np.linalg.norm(v1[:, 3]) * np.linalg.norm(v2[:, 3]))
cosine_similarity_pet = np.dot(v1[:, 4], v2[:, 4]) /
 (np.linalg.norm(v1[:, 4]) * np.linalg.norm(v2[:, 4]))

# Weight the cosine similarity scores by the absolute rankings of
#  the dimensions in the two vectors
weighted_cosine_similarity_animal = (rank(0, v1) + rank(0, v2)) ** -1
weighted_cosine_similarity_mammal = (rank(1, v1) + rank(1, v2)) ** -1
weighted_cosine_similarity_feline = (rank(2, v1) + rank(2, v2)) ** -1
weighted_cosine_similarity_domestic = (rank(3, v1) + rank(3, v2)) ** -1
weighted_cosine_similarity_pet = (rank(4, v1) + rank(4, v2)) ** -1

# Calculate the weighted overlap score
weighted_overlap = np.sqrt(
    np.sum(
        weighted_cosine_similarity_d
        for d in [cosine_similarity_animal, cosine_similarity_mammal,
                  cosine_similarity_feline,
                  cosine_similarity_domestic, cosine_similarity_pet]
    )
    / np.sum((2 * i) ** -1 for i in range(len(v1)))
)

print(weighted_overlap)
```

Figure C.2: Python code snippets on weighted overlap measure for concept “cat”.

Appendix D

Nomenclature

Nomenclature

- WSD: Word Sense Disambiguation
- MLP: Multilayer Perceptron
- WO: Weighted Overlap measure
- $k - NN$: k-Nearest Neighbor approach
- m: mention or the ambiguous word/phrase
- S_i: i-th sense of a mention
- $R(m)$: Representation of mention m
- PD: Paragraph of Document
- PW: Paragraph of Wikipedia page
- $sim(x, y)$: Similarity of vectors x and y
- k: Number of candidates
- T: Input text
- E: Set of possible answers
- E^*: Set of correct answers
- W_i: i-th window
- OFF: Offset size

- π^m: Micro-averaged precision
- π^M: Macro-averaged precision