Instituto Politécnico Nacional

Centro de Investigación en Computación

Plagiarism Detection Through Paraphrase Recognition

Tesis que para obtener el grado de:

Doctorado en Ciencias de la Computación

Presenta:

M. en C. Miguel Ángel Sánchez Pérez

Directores de tesis:

Dr. Alexander Gelbukh

Dr. Grigori Sidorov

Junio, 2018

Ciudad de México, México

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 14:00 horas del día 08 del mes de mayo de 2018 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Plagiarism detection through paraphrase recognition"

Presentada por el alumno:

SÁNCHEZ	**PÉREZ**	**MIGUEL ÁNGEL**
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	1	4	0	5	6	5

aspirante de: **DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA
Directores de Tesis

[illegible]
Dr. Alexander Gelbukh

[illegible]
Dr. Grigori Sidorov

[illegible]
Dr. Sergio Suárez Guerra

[illegible]
Dr. Ildar Batyrshin

[illegible]
Dra. Olga Kolesnikova

[illegible]
Dr. Miguel Jesús Torres Ruiz

PRESIDENTE DEL COLEGIO DE PROFESORES

[illegible]
Dr. Marco Antonio Ramírez Salinas

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

INSTITUTO POLITÉECNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 28 del mes mayo del año 2018, el (la) que suscribe Miguel Ángel Sánchez Pérez alumno (a) del Programa de Doctorado en Ciencias de la Computación con número de registro B140565, adscrito a Centro de Investigación en Computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. Alexander Gelbukh y Dr. Grigori Sidorov y cede los derechos del trabajo intitulado Plagiarism Detection Through Paraphrase Recognition, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección miguel.sanchez.nan@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

[illegible]

Miguel Ángel Sánchez Pérez

Nombre y firma

Resumen

La detección de plagio se puede abordar de dos maneras diferentes: detección intrínseca la cual consiste en, dado un documento, encontrar cambios en el estilo de escritura; y detección externa la cual consiste en recuperar los posibles documentos fuentes y en encontrar los fragmentos que fueron plagiados. La detección de plagio es un tema muy estudiado actualmente y de gran importancia para la comunidad de editores, investigadores e instituciones educativas.

Detectar plagio requiere analizar los diferentes tipos de ofuscación usados en un documento sospechoso, donde la mayoría están estrechamente relacionados al uso de paráfrasis en diferentes niveles. En esta tesis nos enfocamos en encontrar los fragmentos de texto exactos que fueron plagiados, una tarea también conocida como alineación de textos, y en estudiar de forma independiente la tarea de paráfrasis dada su importancia para la detección de plagio.

Primero, modificamos el sistema de plagio que propusimos en trabajos anteriores adaptándola a diferentes tipos de ofuscación. Luego, proponemos un algoritmo genético para optimizar el conjunto de parámetros de nuestro sistema. Ambas aproximaciones mejoraron los resultados del estado del arte.

Con respecto a la identificación de paráfrasis, estudiamos los métodos propuestos en esta tarea basados en bases de conocimientos y en modificaciones del tradicional modelo de espacio vectorial, y proponemos nuevas combinaciones de técnicas en diferentes etapas del proceso como lo son el preprocesamiento, normalización, similitud término-término, similitud de frases, entre otras. Superamos los resultados de los métodos existentes basados en bases de conocimientos.

Finalmente, llevamos a cabo un estudio a fondo de métodos del estado del arte basados en aprendizaje profundo. Estudiamos el desempeño a través de una comparación de la precisión durante el aprendizaje y la evaluación,

así como modificamos ciertos detalles faltantes no descritos en sus modelos. También, proporcionamos información relevante sobre los retos y problemas que presentan dichas aproximaciones en la tarea de identificación de paráfrasis.

Abstract

Plagiarism Detection can be addressed in two different ways: intrinsic detection which consists in, given a document, finding writing style changes; and external detection which consists in retrieving possible source documents and finding the exact fragments that were plagiarized. Plagiarism detection is a hot topic and of paramount importance for publishers, researchers, and educational institutions.

Detecting plagiarism requires analyzing the several types of obfuscation that might be presented in the suspicious documents, most of them closely related to paraphrase on different levels. In this thesis, we focus on finding the exact plagiarized text passages, also known as Text Alignment, and we independently study the paraphrase task given its importance to plagiarism detection.

First, we modify our previously proposed plagiarism detection system to adapt to different types of obfuscation. Then, we propose a genetic algorithm to optimize the set of parameters of our system. Both approaches improved the state-of-the-art results on plagiarism detection.

In the paraphrase regard, we study methods for solving this task based on knowledge bases and modifications of the traditional Space Vector Model and propose new combinations of techniques on different stages of the process like pre-processing, normalization, word-to-word and phrase similarities, among others. We outperformed the existing approaches based on knowledge bases.

Finally, we carry out an in-depth analysis of state-of-the-art approaches using Deep Learning. We study their performance through a comparison of training vs validation accuracy as well as modifying certain details missing from their models. We provided insights into the challenges and issues of Deep Learning approaches for paraphrase identification.

Acknowledgments

First and foremost, I want to express my sincere appreciation to my supervisors, Dr. Alexander Gelbukh and Dr. Grigori Sidorov, for their guidance throughout the course of this work. I thank them for all the time devoted to improving my research and for sharing their extensive knowledge. I am deeply grateful for their unconditional support to every project and collaboration I decided to carry out.

I also thank my exceptional internship advisors, Dr. Efstathios Stamatatos, Dr. Paolo Rosso, Dr. Erik Cambria and Erik Peterson, for their hospitality, support, and the opportunity to collaborate with their teams. It was a great privilege to live, work and share with such a diverse group of people and in such amazing places.

I sincerely thank Helena Gomez, Ilia Markov, my friends, and colleagues, for their words of encouragement, kindness, and adventures lived together.

My deepest gratitude to Centro de Investigación en Computación, Instituto Politécnico Nacional, and CONACyT, for providing the support, opportunities and perfect environment to carry out this work.

Finally, the most special gratitude and love to my family, especially my parents and brother, for their sacrifice, inspiration, unwavering support, and endless love, throughout my whole life. Without their presence and influence, this work and who I am today would not exist.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Plagiarism Detection

There are several definitions of plagiarism. According to Merriam-Webster online dictionary, to "plagiarize" means[1]:

- To steal and pass off (the ideas or words of another) as one's own: use (another's production) without crediting the source.
- Intransitive verb: To commit literary theft: present as new and original an idea or product derived from an existing source.

An interesting article[2] in the www.plagiarism.org portal provides an extended description of what constitutes plagiarism. The author lists a set of actions that are considered plagiarism:

- turning in someone else's work as your own
- copying words or ideas from someone else without giving credit
- failing to put a quotation in quotation marks
- giving incorrect information about the source of a quotation
- changing words but copying the sentence structure of a source without giving credit
- copying so many words or ideas from a source that it makes up the majority of your work, whether you give credit or not

Plagiarism detection and paraphrase identification as an implicit part of the former are hot topics for publishers, researchers, and educational institutions. Paraphrase identification is used in several other tasks beside plagiarism detection, like machine translation [45], information retrieval [55], question answering [7], among others.

In computer science, plagiarism detection is a natural language processing task that can be addressed in two different ways: intrinsic analysis, which aims to identify potential cases of plagiarism searching for writing style changes; and corpus-based analysis, which compares a suspicious document to a collection of possible sources. The later can be divided into two

[1]`https://www.merriam-webster.com/dictionary/plagiarize`
[2]`http://www.plagiarism.org/article/what-is-plagiarism`

subtasks which the PAN workshop series[3] define as Source Retrieval and Text Alignment respectively. Their description goes as follows:

Source Retrieval: Given a suspicious document and a web search engine, the task is to retrieve all plagiarized sources while minimizing the retrieval costs.

Text Alignment: Given a pair of documents, the task is to identify all contiguous maximal-length passages of reused text between them.

We focus on advancing the state-of-art on plagiarism detection, specifically the task of Text Alignment. To achieve this goal, a more granular description of plagiarism is needed, classifying it into different subclasses given the type of obfuscation and approaching its detection accordingly. For example, detecting plagiarism where the copy was verbatim, paraphrased or summarized, needs to be done consequently to the features of each obfuscation type, which at the same time suggests using other parameter settings or applying other methods.

Addressing these problems brings two main issues into the spotlight. First, improving or proposing new ways to detect plagiarism on the document level. Then, recognizing paraphrase, the most common phenomena in plagiarized documents. Detectors at the document level, like those presented at PAN workshops, usually focused on basic lexical and syntactic similarities due to their simplicity and computational efficiency. In the other hand, paraphrase recognition is approached as a classification problem at the phrase or sentence level only given its complexity and computational cost, and it is commonly referred as paraphrase identification. All of this, gives room to propose new and innovative models to resolve both tasks and integrating them efficiently into a plagiarism detection system.

Hence, our efforts are directed to contribute solving these issues. First, by improving our previously proposed method at the PAN Text Alignment task [73], and then applying new techniques to the paraphrase identification. For the Text Alignment objective, we plan to use machine learning to integrate various methods and to optimize our model's parameters, while for paraphrase recognition we plan to apply new ways of computing similarities like the soft cosine and more advance techniques based on Deep Learning.

[3]`https://pan.webis.de`

1.2 Hypothesis

The hypotheses raised in this work are:

1. Adapting the model to handle different types of obfuscation through different parameters and/or different methods improves performance.

2. A genetic algorithm helps to find a near optimal set of parameters.

3. A combination of methods works better than each one separately in paraphrase identification.

4. Deep learning only works when there is much data or some sort of transfer learning for models trained on other datasets and/or tasks.

1.3 Objectives

1.3.1 General Objective

To build an optimized and adaptable computational model for plagiarism detection, capable of recognizing different types of obfuscation, focused on paraphrase identification.

1.3.2 Particular Objectives

1. Develop a plagiarism detection model adaptive to different types of obfuscation.
2. Optimize the parameters of a plagiarism detection model and analyze their behavior on various obfuscation types.
3. Study and experiment with new approaches for paraphrase identification.

1.4 Main Contributions

We divide the main contributions of our work into scientific and practical. In the former, we state our proposed models and experiments that have been published in journals or conference articles, or that are in the process of being published. In the later we refer to the software, tools and resources we have created that are freely available to the scientific community.

1.4.1 Scientific Contributions

- Adaptive plagiarism model per obfuscation type with state-of-the-art results.
- Optimization model to improve the state-of-the-art results that can be use in a traditional or paraphrase-focused plagiarism detection model.
- Combination of knowledge-based models for paraphrase identification that outperforms similar approaches.
- Exploration of deep learning models and in-depth analysis of its performance and reproducibility challenges.

1.4.2 Practical Contributions

- Code in Python of our genetic model for parameter tuning of our plagiarism detection system.
- Jupyter notebook using Python for easy testing of WordNet similarity metrics applied to the paraphrase identification task.
- Three deep learning models implementations for paraphrase identification using Python and Keras capable of running on GPU and allowing fast prototyping of new models and experiment designs.

1.5 Structure of the Document

Chapter 1 details the problems, motivation, hypotheses, and contributions of this work. Chapter 2 describes methods and models as part of the theoretical background necessary to understand the content of this thesis. Chapter 3 describes related work in both plagiarism detection and paraphrase identification tasks. Chapter 4 contains the proposed models, experiments and results on the text alignment task. Specifically, the adaptive model to different obfuscation types and the genetic algorithm to optimize the plagiarism detection parameters. Chapter 5 includes the proposed model for knowledge base paraphrase identification and a detailed analysis of selected approaches based on deep learning. Chapter 6 draws conclusions and provides directions for future work.

Chapter 2

Theoretical Framework

2.1 Basic Text Processing Techniques

2.1.1 Stop Words

We place words in different categories depending on their grammatical functions, which in turn can be open or closed classes. An open class is one that commonly accepts the addition of new words, while a close class is one to which new items are rarely added.

Stop words, stop list, function words; usually, refers to the most common words in a language. They are deemed unlikely to be useful for searching in information retrieval system based on a word-by-word match or for computing similarity in a vector space models. Most of these words belong to closed classes like articles, prepositions, pronouns, auxiliary verbs, etc. A small stop list extracted from the British National Corpus is shown in Table 2.1.

The usefulness of removing the stop words depends greatly on the task being solved and the taken approach. For instance, in plagiarism detection, Stamatatos [80] demonstrated that stop words n-grams can capture syntactic similarities between a suspicious and original documents and that they can be used to detect the exact plagiarized passage boundaries. Moreover, he showed the robustness of this approach when dealing with highly obfuscated cases where most of the words or phrases have been replaced with synonyms. Also, some search engines avoid using these stop lists because it prevents from searching phrases that contain stop words like *when and where.*

In the other hand, some authors in order to improve performance prefer to remove stop words. For instance, in the plagiarism detection task, Kong's approach [32] is based on a bag of words model and cosine similarity where the order of words is not relevant, and hence, phrases are not taken into consideration; also, Torrejón [69] uses context skip n-grams formed by only important words. In text summarization, Yulia [36] applied this technique reducing the content of the text to more specific expressions (multi-word descriptions), containing only the words that are useful and meaningful for the generation of automatic summaries. Additionally, a stop list has the advantage that it reduces the size of the inverted index. According to Zipf's law, a stop list that covers a few dozen words can reduce the size of the inverted index by half [41].

Table 2.1: List of the 50 most frequent words of BNC corpus

1. the	11. with	21. are	31. or	41. her
2. of	12. he	22. not	32. an	42. n't
3. and	13. be	23. his	33. were	43. there
4. a	14. on	24. this	34. we	44. can
5. in	15. i	25. from	35. their	45. all
6. to	16. that	26. but	36. been	46. as
7. is	17. by	27. had	37. has	47. if
8. was	18. at	28. which	38. have	48. who
9. it	19. you	29. she	39. will	49. what
10. for	20. ‘s	30. they	40. would	50. said

2.1.2 Morphology

We may have different forms of a word given the tense, number or gender of it. For example, *sit*, *sits*, or *sat* represents the same word *sit*, which raises the question of separating or collapsing them depending on the task we choose. When grouping such forms together and working concerning lexemes, we have some basic techniques like lemmatization or stemming.

Stemming

Stemming refers to a simplified form of morphological analysis consisting merely of truncating a word. For example, *laughing*, *laugh*, *laughs*, and *laughed* are all stemmed to *laugh-*. Common stemmers are the Lovins and Porter stemmers, which differ in the actual algorithms used for determining where to truncate words [39, 60].

Two problems with the truncation stemmers are that they conflate semantically different words (for example, *gallery* and *gall* may both be stemmed to *gall-*) and that the truncated stems can be unintelligible to users. They are also much harder to make work well for morphology-rich languages [41].

Lemmatization

Lemmatization consists in attempting to find the lemma or lexeme of an inflected form. This process implies disambiguation at the level of lexemes, such as whether the use of *lying* represents the verb *lie-lay* meaning ‘to prostrate oneself’, or *lie-lied* meaning ‘saying an intentionally false statement’.

2.2 Part of Speech Tagging

The task of part of speech (POS) tagging is the process of assigning a grammatical tag to each word in a phrase or sentence. Tagging a word involve taking into consideration its definition and its context and is usually addressed in one of two ways: rule-based algorithms or probabilistic models. Most successful approaches achieve quite high accuracy between 96% and 97%.

When tagging tokens in a sentence, some are processed given its definition, for example those that always have the same tag like in the case of articles. However, to tag some words we need to look more into its context. For example, the word "play" can be considered as a noun (a) or as a verb (b).

(a) That was a good play.

(b) He likes to play.

Several authors have created sets of tags with different levels of granularity. One of the most populars is the Penn tag set for English shown in Table 2.2.

Describing methods for POS tagging is out of the scope of this thesis but we strongly recommend reviewing some of the basic methods presented by Manning and Schütze in their book *Foundations of Statistical Natural Language Processing* [41].

Table 2.2: Penn POS tags for English

Tag	Part Of Speech
AT	article
BEZ	the word is
IN	preposition
JJ	adjective
JJR	comparative adjective
MD	modal
NN	singular or mass noun
NNP	singular proper noun
NNS	plural noun
PERIOD	. : ? !
PN	personal pronoun
RB	adverb
RBR	comparative adverb
TO	the word to
VB	verb, base form
VBD	verb, past tense
VBG	verb, present participle, gerund
VBN	verb, past participle
VBP	verb, non-3rd person singular present
VBZ	verb, 3rd singular present
WDT	wh- determiner (what, which)

2.3 Vector Space Model

The vector space model is a traditional and widely used model in natural language processing tasks and was first popularized in information retrieval. The main characteristic that makes this model so popular is its simplicity and the possibility of computing semantic similarity using a traditional vector space. The model consists in representing text documents in a high-dimensional space where each dimension is a word from a vocabulary in the document collection selected.

This model is also regarded as Bag of Words (BOW) since the order of the dimensions of the vectors does not matter when defining the space.

Defined the dimensions of the space, representing documents on this space requires selecting a weighting scheme, meaning giving a value to each dimension.

2.3.1 Weighting Schemes

There are several schemes that have been proposed, the simplest being a binary representation, i.e., Assigning 1 to a dimension (word) if the word occurs in the document, 0 otherwise. We could also use the count of a word in the document, called term frequency $tf_{i,j}$ which is defined as the number of occurrences of word w_i in document d_j. Through experimentation, researchers have acknowledge the need to apply a smoothing non-linear function to prevent higher values of co-occurrences from receiving too much importance, for example $log(tf_{i,j})$.

Term frequency assumes that a higher occurrence of word describes better the content of a document. However, this assumption is not always true, for example, the words we identified as stop words in 2.1.1, are words that will occur quite often in every single document and that do not significantly contribute to the meaning of a text.

Another metric often use is the inverse document frequency $idf(w, D)$ which is the ratio of the total of documents in our collection and the number of documents where a word occurs. This metric is an indicator of the specificity of a term in a given corpus, for example terms related to a particular topic in a collection where we have plenty of topics. This metric will reduce drastically the importance of stop words since these occurs in most of the documents regardless of the topic. Logarithm is also applied to this metric to reduce the importance of rare term in large corpora.

The most effective method for weighting uses a combination of term frequency and inverse document frequency $tf \cdot idf$.

2.3.2 Vector Similarity

There are several vector similarity measures used in natural language processing that can be interpreted as semantic similarity. When using a binary weighting scheme we can define some of these measures. Table 2.3.2, extracted from [41], lists some of the most common measures.

Table 2.3: Similarity measures for binary vectors

Similarity measure	Definition
matching coefficient	$\|X \cap Y\|$
Dice coefficient	$\frac{2\|X \cap Y\|}{\|X\|+\|Y\|}$
Jaccard coefficient	$\frac{\|X \cap Y\|}{\|X \cup Y\|}$
Overlap coefficient	$\frac{\|X \cap Y\|}{\min(\|X\|,\|Y\|)}$
cosine	$\frac{\|X \cap Y\|}{\sqrt{\|X\|+\|Y\|}}$

For real-value vectors with have two main measures: cosine (of the angle between two vectors) and Euclidean distance. They are define as follows:

$$cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum\limits_{i=1}^{n} x_i y_i}{\sqrt{\sum\limits_{i=1}^{n} x_i^2}\sqrt{\sum\limits_{i=1}^{n} y_i^2}},$$

$$euc(\vec{x},\vec{y}) = |\vec{x}-\vec{y}| = \sqrt{\sum_{i=1}^{n}(x_i i y_i)^2}$$

Semantic Similarity

One of the properties of the traditional space vector is orthogonality, meaning that all of its dimensions are perpendicular to each other. In other words, we assume that each word in our vocabulary is completely different the rest. This assumption is obviously wrong given that in languages we have synonyms or closely related words. This issue suppose a weakness of the traditional vector space model to compute semantic similarity where the text have been paraphrased.

There are some approaches that take into account relations between dimensions of the vector space. Usually these approaches make use of knowledge bases to compute term-to-term similarities and composed them into a phrase similarity metric. One of the most popular of such bases is WordNet [51] which we described in section 2.4.

Mihalcea et al. [47] propose a model to compute the similarity between two phrases using term-to-term similarity. Given two text phrases T_1 and T_2 the proposed scoring function is:

$$sim(T_1,T_2) = \frac{1}{2}\left(\frac{\sum_{w\in\{T_1\}}(maxSim(w,T_2)*idf(w))}{\sum_{w\in\{T_1\}} idf(w)} + \frac{\sum_{w\in\{T_2\}}(maxSim(w,T_1)*idf(w))}{\sum_{w\in\{T_2\}} idf(w)}\right), \tag{2.1}$$

where $idf(w)$ represents the inverse document frequency of w in the British National Corpus (BNC) and $maxSim(w,T_2)$ is the maximum similarity of word w to all words in T_2 using a WordNet similarity metric.

Fernando and Stevenson [17] use binary representations of sentences and compute a matrix of word-to-word similarities W of all the words in the vocabulary computed using WordNet metrics. Assuming $\vec{a}$ and $\vec{b}$ are the binary vectors of two sentences then their similarity is defined as:

$$sim(\vec{a},\vec{b}) = \frac{\vec{a}W\vec{b}^T}{|\vec{a}||\vec{b}|} \tag{2.2}$$

Similarly, Sidorov et al. [77] uses a word-to-word similarity matrix to complement their metric. The called their model SoftCosine where its rational is breaking the orthogonality of the traditional vector space model. It is defined as:

$$soft\_cosine_1(\vec{a},\vec{b}) = \frac{\sum\limits_{i,j}^{N} s_{ij}a_ib_j}{\sqrt{\sum\limits_{i,j}^{N} s_{ij}a_ia_j}\sqrt{\sum\limits_{i,j}^{N} s_{ij}b_ib_j}}, \tag{2.3}$$

where s_{ij} is an element of the word-to-word similarity matrix W. The difference between the last two approaches is that Sidorov et al. uses real-valued vectors and normalizes the metric using the values of the similarity matrix as well.

2.4 WordNet

Quoting the resource website[1], *WordNet®is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations.*

Most of WordNet's relations are between same part of sppech (POS), either nouns, verbs, adjectives and adverbs. Thus, we can see it as four different sub-nets. This should be taken into consideration when computing word-to-word similarities.

As mentioned before, the main unit in WordNet is the synset which comprise words that are synonyms. It also includes a description of each word and in some cases example sentences.

WordNet builds as an ontology where the main relation y the is_a, also called hyperonymy, hyponymy or super-subordinate. For example: car is a type of vehicle. This structure tree-liked is the base of the similarity metrics we will see in the next section.

WordNet is included in NLTK Python module along with the most popular similarity measures. This allows to easily implement approaches that include knowledge-based semantic similarity.

2.4.1 Wordnet Similarity Measures

There are several metrics that works using the WordNet ontology and in some cases a supporting corpus.

Patwardhan [57] and Mihalcea [47] use these metrics for Word Sense Disambiguation and Paraphrase Identification. They provide a brief description of these measures as follows:

Leacock-Chodorow

The measure of Leacock-Chodorow [35] uses the length of the shortest path between two concepts $shortest\_length(c_1, c_2)$ and the maximum depth D of the WordNet ontology.

$$sim_{lch}(c_1, c_2) = -\log \frac{shortest\_length(c_1, c_2)}{2 * D}. \tag{2.4}$$

[1]`https://wordnet.princeton.edu/`

Wu and Palmer

The measure of Wu and Palmer [86] uses the depth of each concept and the depth of the closest ancestor shared by both concepts, also referred as the least common subsumer (LCS).

$$sim_{wup}(c_1, c_2) = \frac{2 * depth(LCS(c_1, c_2))}{depth(c_1) + depth(c_2)}. \tag{2.5}$$

Resnik

The measure of Resnik [68] uses the Information Content (IC) of the least common subsumer (LCS). The Information Content measures the specificity of a concept. It depends on frequency counts of words in a corpus where each occurrence of a word also affects the counts of all of its ancestors in the WordNet taxonomy. Given the structure of WordNet taxonomy, it is only possible to compute the IC of nouns and verbs. From the frequency counts IC is defined as:

$$IC(c) = -\log P(c). \tag{2.6}$$

Hence, higher values are associated with more specific concepts (e.q. car), while more general concepts receive lower values (e.q. vehicle). Defined the IC of a concept, the measure of Resnik is defined as:

$$sim_{res}(c_1, c_2) = IC(LCS(c_1, c_2)). \tag{2.7}$$

Lin

The measure of Lin [38] also uses the Information Content metric and it is defined as:

$$sim_{lin}(c_1, c_2) = \frac{2 * IC(LCS(c_1, c_2))}{IC(c_1) + IC(c_2)}. \tag{2.8}$$

2.4.2 Jiang & Conrath

Similarly, the measure of Jiang & Conrath combine the Information Content metric and least common subsumer in the following score:

$$sim_{jcn} = \frac{1}{IC(c_1) + IC(c_2) - 2 * IC(LCS(c_1, c_2))} \tag{2.9}$$

2.5 Deep Learning Architectures

2.5.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are widely popular in computer vision and have been successfully applied in tasks like facial recognition, self-driving cars, hand-written recognition, among others. Most recently there have been used in a series of natural language processing tasks, specially classification problems, like author profiling [78], personality detection [40], paraphrase identification [26], sentence modelling [30], sentiment analysis [74], and others.

As its name says, CNN works by convolving a function (filter) over another (input). In other words, it is a sliding window function applied to an input matrix.

Formally, given k filters the output of a CNN is defined as:

$$f_{ij}^k = g((W^k * x)_{ij} + b^k), \tag{2.10}$$

where g is a non-linear function, W^k and b^k are the weights and bias representing each filter k to be learned.

Although CNN refers only to a convolution operation, when talking about these type of networks, the expected architecture usually includes an input, convolution, pooling and fully-connected layers.

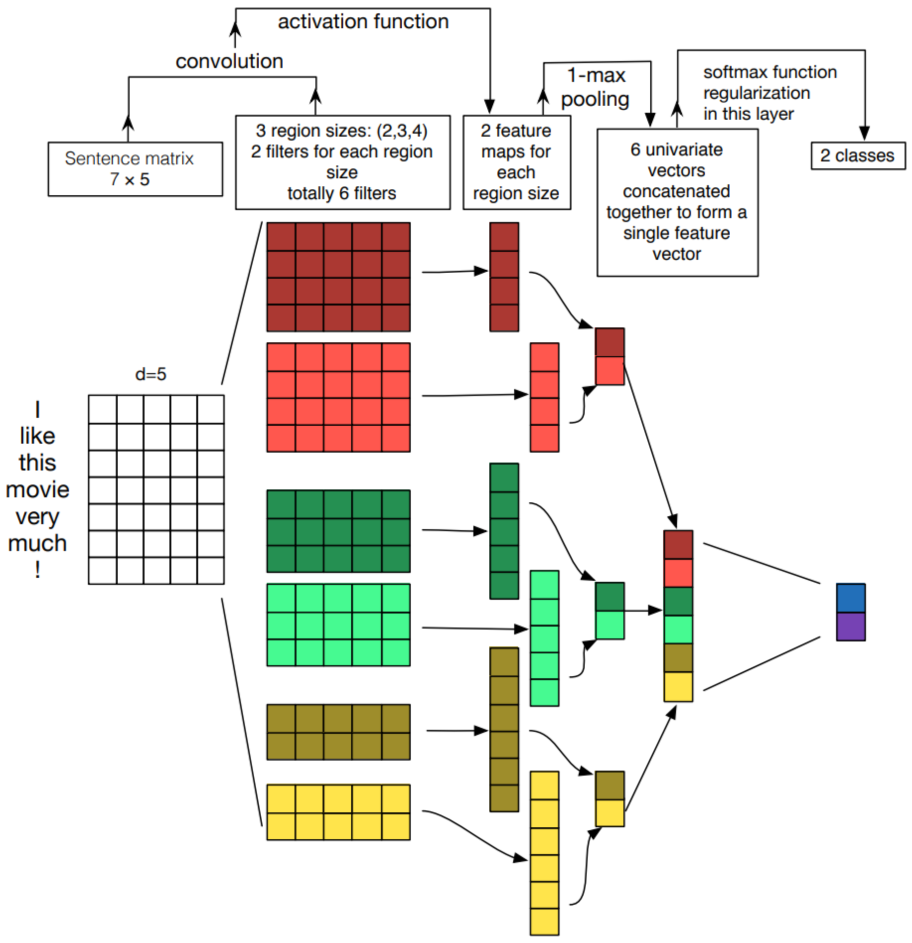
In Figure 2.5.1 we show an example of CNN extracted from [88]. The model is applied to a sentence where each word is represented as word embeddings.

2.5.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN) or sequence models more generally are very useful when dealing with sequential data like text, audio or video. There is a set of problems that have been addressed successfully using these models like speech recognition, sentiment classification, machine translation, among others.

These neural networks receive their “recurrent” name because they perform the same set of operations on every element of the sequence. The main property of RNNs is that they are able to memorize information it has seen before in the sequence. In Figure 2.5.2 we show the basic diagram of a RNN.

Figure 2.1: CNN applied to text


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

There are several configurations of RNNs depending on the task at hand and its expected input and output sizes . In Figure 2.5.2 we show the possible configurations.

For example, Part-of-Speech tagging models output a tag for each input word (many-to-many), sentiment analysis returns a class given a sequence of words (many-to-one), a text generator creates a sequence from an input word (one-to-may) , and in machine translation a sequence of words is translated to another after the first was fully processed (second type of many-to-many).

The diagram of a cell of a basic RNN is given in Figure 2.5.2.

Figure 2.2: Basic RNN diagram

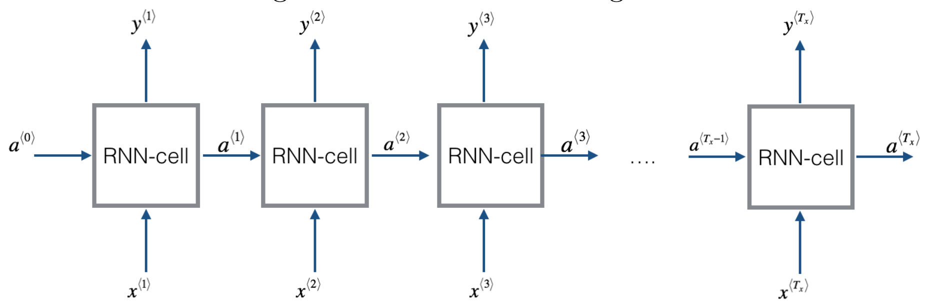
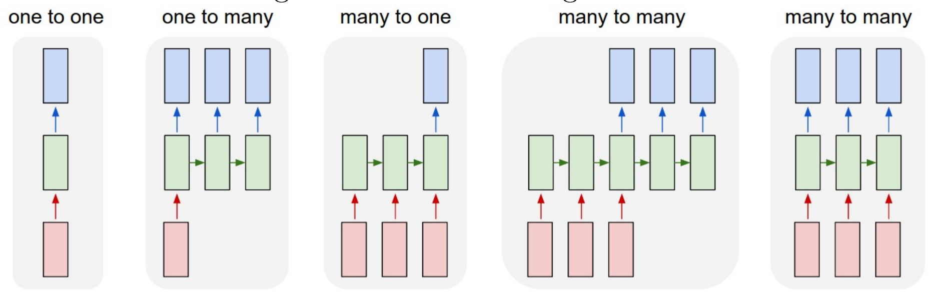


Figure 2.3: RNNs configurations



Its activations and output values is defined as:

$$\begin{aligned} a^{<t>} &= \tanh(W_{ax}x^{<t>} + W_{aa}a^{<t-1>} + b_a) \\ \hat{y}^{<t>} &= softmax(W_{ya}a^{<t>} + b_y) \end{aligned} \tag{2.11}$$

In practice, basic RNN can look back only a few steps and therefore they are not very effective to model long term dependencies. Also, when dealing with long sequences this models suffer from vanishing gradient problem. There are other architectures of RNNs other than the basic that addressed this problems: GRU and LSTM.

Long Short-Term Memory

In this section we briefly describe one of the most popular RNN architectures known as Long Short-Term Memory (LSTM). It follows the same principles as the basic RNN changing the definition of the cell as shown in Figure 2.5.2.

Figure 2.4: Basic RNN cell

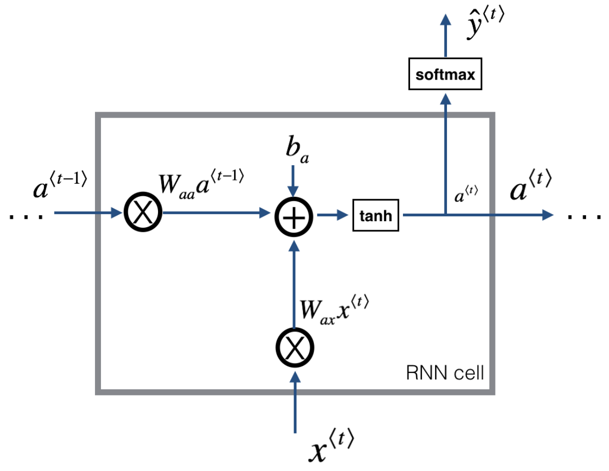
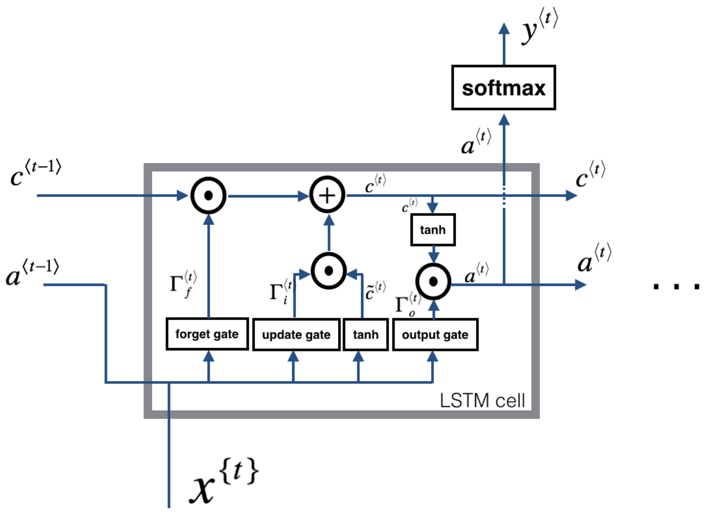

$\hat{y}^{\langle t \rangle}$
softmax
b_a
$a^{\langle t-1 \rangle}$
$W_{aa}a^{\langle t-1 \rangle}$
tanh
$a^{\langle t \rangle}$
$a^{\langle t \rangle}$
$W_{ax}x^{\langle t \rangle}$
RNN cell
$x^{\langle t \rangle}$

Figure 2.5: LSTM cell



Its activations, states, and outputs are defined as:

$$\begin{aligned}
\Gamma_f^{<t>} &= \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \\
\Gamma_u^{<t>} &= \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \\
\tilde{c}^{<t>} &= \tanh(W_C[a^{<t-1>}, x^{<t>}] + b_C) \\
c^{<t>} &= \Gamma_f^{<t>} \circ c^{<t-1>} + \Gamma_u^{<t>} \circ \tilde{c}^{<t>} \\
\Gamma_o^{<t>} &= \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \\
a^{<t>} &= \Gamma_o^{<t>} \circ \tanh(c^{<t>}),
\end{aligned} \tag{2.12}$$

where $\circ$ is the element-wise multiplication.

2.6 Our Approach to Text Alignment at PAN

We describe our approach in the paper *Adaptive Algorithm for Plagiarism Detection: The Best-performing Approach at PAN 2014 Text Alignment Competition* [71], using the three steps model: seeding, extension, and filtering. Before going through these steps, we pre-process the documents taking the following actions:

- sentence splitting and tokenization,
- removing all tokens (“words”) that do not start with a letter or digit,
- reducing all letters to lowercase,
- applying stemming,
- joining short sentences (shorter than $minsentlen = 3$ words) with the next one (if the new joint “sentence” was still short, we join it with the next one, etc.).

In the subsequent sections, we describe our processes of seeding, extension, and filtering.

2.6.1 Seeding

Given a suspicious document and a source document, the task of the seeding stage is to construct a large set S of similar short passages called *seeds*. Each seed is a pair that consists of a small fragment of the suspicious document and a small fragment of the source document that are in some sense similar. In our case, the fragments to form the pairs were sentences, which may be joined as described above. Constructing these pairs required to measure the similarity between sentence vectors, for which we had to choose a weighting scheme.

To compute the similarity between two sentences, we represented individual sentences with a tf-idf vector space model (VSM), as if each sentence were, in the terminology of VSM, a separate “document,” and all sentences in the pair of the original document formed a “document collection.” The idf measure calculated in this way is called *isf measure* (inverse sentence frequency) to emphasize that it is calculated over sentences as units and not documents:

$$\begin{aligned} tf\left(t, s\right) &= f\left(t, s\right), \\ isf\left(t, D\right) &= \log \frac{|D|}{|\{s \in D : t \in s\}|}, \\ w\left(t, s\right) &= tf\left(t, s\right) \times isf\left(t, D\right), \end{aligned}$$

where for term frequency $tf(t, s)$ we simply used the number of occurrences $f(t, s)$ of the term t in the sentence s; D is the set of all sentences in both given documents, and $w(t, s)$ is the final weight of a term t of the sentence s in our VSM representation.

After we defined the weighting scheme and transformed all sentences into vectors in both documents, we compared each sentence in the suspicious document to each sentence in the source document.

Now, we construct the desired set S as

$$S = \{(i, j) \mid \cos\left(susp_i, src_j\right) > mincos \wedge dice\left(susp_i, src_j\right) > mindice\},$$

where the two sentences are represented as vectors, cos is the cosine similarity, *dice* is the Dice coefficient:

$$\begin{aligned} \cos\left(susp_i, src_j\right) &= \frac{susp_i \cdot src_j}{|susp_i| \times |src_j|}, \\ dice\left(susp_i, src_j\right) &= \frac{2\left|\delta\left(susp_i\right) \cap \delta\left(src_j\right)\right|}{\left|\delta\left(susp_i\right)\right| + \left|\delta\left(src_j\right)\right|}, \end{aligned}$$

$\delta(x)$ is the set of non-zero coordinates of a vector x, $|*|$ is the Euclidean length of a vector or the cardinality of a set, respectively, and *mincos* and *mindice* are some thresholds determined experimentally.

2.6.2 Extension

Given the set of seeds S, defined as the pairs (i, j) of similar sentences, the task of the extension stage is to form larger text fragments that are similar between two documents. For this, the sentences i are joint into maximal contiguous fragments of the suspicious document, and sentences j into maximal contiguous fragments of the source document, so that those large fragments be still similar.

We divide the extension process into two steps: (1) clustering and (2) validation. In the clustering step, we create text fragments grouping seeds that are not separated by more than a *gap* number of sentences. In our implementation, an easier way to proceed is to sort and cluster the set of seeds by i (left or suspicious document) such that $i_n - i_{n+1} \leq susp\_gap$. Then, for each of the resulting clusters, sort and cluster by j using a *src_gap* threshold (right or source document), and thereby alternate by i and j until no new clusters are formed. Each cluster should have at least *minsize* seeds or will be discarded. Since we use the parameters *susp_gap* and *src_gap* to cluster seeds into larger text fragments, some sentences in these fragments may have no similarity to any of the sentences in the corresponding fragment. Therefore to avoid adding to much noise in the clustering step we validate that the similarity between the text fragments of the remaining clusters exceeds some threshold. If the similarity is less than the given threshold, we apply the extension stage using $susp\_gap - 1$ and $src\_gap - 1$ for this particular cluster. We will reduce the gaps at most to a *min_susp_gap* and *min_src_gap* values, respectively. If the any of the minimum values is reached and the validation condition is not met, then the cluster is discarded.

A text fragment is defined as the collection of all the sentences among the seeds of a particular cluster. Given a cluster integrated by seeds of the form (i, j), then the text fragment in the suspicious document F_{susp} is the collection of all the sentences from the smallest i to the largest i in the cluster. Similarly, the corresponding text fragment in the source document F_{src} is the collection of all the sentences from the smallest j to the largest j in the cluster.

We measured the similarity between text fragments F_{susp} and F_{src} computing the cosine between their vectors:

$$similarity\left(F_{susp}, F_{src}\right) = \cos\left(\sum_{v \in F_{susp}} v, \sum_{v \in F_{src}} v\right), \tag{2.13}$$

where the vector representation of the fragments is done adding together the vectors corresponding to all sentences of F_{susp} and F_{src} respectively.

For details of our method, see Algorithm 1. The variable *side* indicates by which side the pairs are clustered: $+1$ means clustering by sentences of the suspicious document (i) and -1, by sentences of the source document (j). In the algorithm, we generalize the thresholds as *maxgap* and *minsize* but in the implementation, there are separate thresholds for each document, and they are used depending in which side we are clustering. The output of the

Algorithm 1: Extension algorithm

```
   const $minsize$, $minsim$
   Function extension($seeds, maxgap$)
1      $clusters \leftarrow$ clustering($seeds$,$maxgap$,+1)
2      $clusters \leftarrow$ validation($clus$,$maxgap$)
3      return $clusters$
   Function clustering($seeds, maxgap, side$)
1      $clusters \leftarrow$ clusters of $seeds$ such that in each cluster, $side$-hand
       sentences form in the document fragments with at most
       $maxgap$-sentence gaps
2      discard all $c \in clusters$ such that $|c| < minsize$
3      if $|clusters| \leq 1$ then
4          return $clusters$
       else
5          $result \leftarrow \emptyset$
6          foreach $c \in clusters$ do
7              $result \leftarrow result \cup$ clustering($c$,$maxgap$,$-side$)
8      return $result$
   Function validation($clusters, maxgap$)
1      $result \leftarrow \emptyset$
2      foreach $c \in clusters$ do
3          if $similarity(F_{susp}(c), F_{src}(c)) < th\_val$ then
4              if $maxgap > min\_maxgap$ then
5                  $result \leftarrow result \cup$ extension($c$,$maxgap - 1$)
           else
6              $result \leftarrow result \cup \{ c \}$
7      return $result$
```

Extension stage is a set of pairs of similar text fragments $\{(F_{susp}, F_{src}), \dots\}$ taken from the resulting clusters.

2.6.3 Filtering

Given the set $\{(F_{susp}, F_{src}), \dots\}$ of plagiarism cases, the task of the filtering stage is to improve precision (at the expense of recall) by removing some "bad" plagiarism cases. We did the filtering in two stages: first, we resolved overlapping fragments; then, we removed too short fragments (in what follows we only refer to fragments that represent plagiarism cases, not to arbitrary fragments of the documents).

Resolving overlapping cases.

We call two plagiarism cases $\left(F_{susp}^{'}, F_{src}^{'}\right)$ and $\left(F_{susp}^{''}, F_{src}^{''}\right)$ overlapping if the fragments $F_{susp}^{'}$ and $F_{susp}^{''}$ share (in the suspicious document) at least one sentence. We assume that the same source fragment can be used several times in a suspicious document, but not vice versa: each sentence can be plagiarized from only one source and thus can only belong to one plagiarism case. To simplify things, instead of re-assigning only the overlapping parts, we simply discarded whole cases that overlapped with other cases. Specifically, we used the following algorithm:

1. While exists a case P ("pivot") that overlaps with some other case
   (a) Denote $\Psi(P)$ be the set of cases $Q \neq P$ overlapping with P
   (b) For each $Q \in \Psi(P)$, compute the quality $q_Q(P)$ and $q_P(Q)$; see (2.14)
   (c) Find the maximum value among all obtained $q_y(x)$
   (d) Discard all cases in $\Psi(P) \cup \{P\}$ except the found x

In our implementation, at the first step, we always used the first case from the beginning of the suspicious document. We compute the quality function $q_y(x)$ of the case x on an overlapping case y as follows. The overlapping cases $x = \left(F_{susp}^{x}, F_{src}^{x}\right)$ and $y = \left(F_{susp}^{y}, F_{src}^{y}\right)$ are pairs of corresponding fragments. Let $O = F_{susp}^{x} \cap F_{susp}^{y}$ be the overlap and $N = F_{susp}^{x}/O$ be the non-overlapping part. Then the quality

$$q_y(x) = sim_{F_{src}^{x}}(O) + \left(1 - sim_{F_{src}^{x}}(O)\right) \times sim_{F_{src}^{x}}(N), \qquad (2.14)$$

where sim is a non-symmetric similarity of a fragment F_{susp} (in the suspicious document) to a reference fragment F_{src} (in the source document):

$$sim_{F_{src}}(F_{susp}) = \frac{1}{|F_{susp}|} \sum_{s \in F_{susp}} \max_{r \in F_{src}} (\cos(s, r)).$$

Formula (2.14) combines the similarity of the overlapping part and the non-overlapping part of the suspicious fragment to the source counterpart.

Table 2.4: Our results on PAN 2013 training and test corpus

Obfuscation	PAN 2013 training corpus			
	Plagdet	Recall	Precision	Granul.
None	0.8938	0.9782	0.8228	1.0000
Random	0.8886	0.8581	0.9213	1.0000
Translation	0.8839	0.8902	0.8777	1.0000
Summary	0.5772	0.4247	0.9941	1.0434
Entire	0.8773	0.8799	0.8774	1.0021
Obfuscation	**PAN 2013 test corpus**			
	Plagdet	Recall	Precision	Granul.
None	0.9003	0.9785	0.8336	1.0000
Random	0.8841	0.8606	0.9101	1.0008
Translation	0.8865	0.8895	0.8846	1.0008
Summary	0.5607	0.4127	0.9991	1.0588
Entire	0.8781	0.8790	0.8816	1.0034

Removing small cases.

We also discard the plagiarism cases that relate small fragments: if either suspicious or source fragment of a case has a length in characters less than *minplaglen*, then the case is discarded.

2.6.4 Results

Potthast et al. introduced the evaluation framework for plagiarism detection on this specific task in [64]. The evaluation framework refers to the Precision, Recall, Granularity and Plagdet measures. We trained our system using the corpus provided for the PAN 2014 competition (pan13-text-alignment-training-corpus-2013-01-21). We also evaluated our model on the test corpus of PAN 2013 (pan13-text-alignment-test-corpus2-2013-01-21) in order to compare our approach with existing approaches. Table 2.4 shows our results on the training corpus of PAN 2014, which was the same as the training corpus of PAN 2013, and on the test corpus of PAN 2013. Table 2.5 compares our results using the cumulative Plagdet measure with those of the systems submitted to PAN 2013. The columns show the system results on each sub-corpus built using different types of obfuscation.

We experimented with each one of our improvements separately and veri-

Table 2.5: Comparative results according to the Plagdet measure on PAN 2013 test corpus. Performance of the systems was published in [63]

Team	**Entire corpus**	None	Random	Translation	Summary
Sanchez-Perez	**0.8781**	0.9003	**0.8841**	**0.8865**	0.5607
Torrejón	0.8222	0.9258	0.7471	0.8511	0.3413
Kong	0.8189	0.8274	0.8228	0.8518	0.4339
Suchomel	0.7448	0.8176	0.7527	0.6754	**0.6101**
Saremi	0.6991	0.8496	0.6566	0.7090	0.1111
Shrestha	0.6955	0.8936	0.6671	0.6271	0.1186
Palkovskii	0.6152	0.8243	0.4995	0.6069	0.0994
Nourian	0.5771	0.9013	0.3507	0.4386	0.1153
Baseline	0.4219	**0.9340**	0.0712	0.1063	0.0446
Gillam	0.4005	0.8588	0.0419	0.0122	0.0021
Jayapal	0.2708	0.3878	0.1814	0.1818	0.0594

fied that they do boost the cumulative Plagdet measure. Both, the use of the tf-isf measure and our recursive extension algorithm, considerably improved recall without a noticeable detriment to precision. On the other hand, resolution of overlapping cases improved precision without considerably affecting recall. Finally, the dynamic adjustment of the gap size improved Plagdet on the summary sub-corpus by 35% without considerably affecting other corpora.

We participate in the Text Alignment task of the PAN 2014 Lab outperforming all 10 participants as shown in Table 2.6. The official results showed that recall is the measure where we excel but need to improve the precision of the model by identifying and adjusting to other types of obfuscation rather than just the summary obfuscation. Regarding the system runtime, even our goal is not aiming at efficiency, out software performed at an average level.

Table 2.6: PAN 2014 official results reported in [62] using TIRA [23]

Team	**PlagDet**	Recall	Precision	Granularity	Runtime
Sanchez-Perez	**0.8781**	**0.8790**	0.8816	1.0034	00:25:35
Oberreuter	0.8693	0.8577	0.8859	1.0036	00:05:31
Palkovskii	0.8680	0.8263	0.9222	1.0058	01:10:04
Glinos	0.8593	0.7933	**0.9625**	1.0169	00:23:13
Shrestha	0.8440	0.8378	0.8590	1.0070	69:51:15
R. Torrejón	0.8295	0.7690	0.9042	1.0027	**00:00:42**
Gross	0.8264	0.7662	0.9327	1.0251	00:03:00
Kong	0.8216	0.8074	0.8400	1.0030	00:05:26
Abnar	0.6722	0.6116	0.7733	1.0224	01:27:00
Alvi	0.6595	0.5506	0.9337	1.0711	00:04:57
Baseline	0.4219	0.3422	0.9293	1.2747	00:30:30
Gillam	0.2830	0.1684	0.8863	**1.0000**	00:00:55

Chapter 3

Related Work

3.1 Plagiarism Detection

Plagiarism detection, and more generally, text reuse detection, has become a hot research topic given the increasing amount of information being produced as the result of easy access to the Web, large databases and telecommunication in general, which poses a serious problem for publishers, researchers, and educational institutions [44]. Plagiarism detection techniques are also useful in applications such as content authoring systems, which offer fast and simple means for adding and editing content and where avoiding content duplication is desired [4]. Hence, detecting text reuse has become imperative in such contexts.

Two main forms of detecting plagiarism exist, intrinsic and external. The former refers to, given a suspicious document, identifying text passages in it which deviate in its style from the remainder of the document. The later stands for, given a suspicious document and a (vast) collection of potential sources, finding text passages in the collection that are highly similar to text passages in the suspicious document.

Figure 3.1 shows the generic retrieval process to detect external plagiarism. The process is divided into three basic steps, which are typically implemented in most plagiarism detectors. First, source retrieval, which identifies a small set of candidate source documents that are likely sources for plagiarism regarding the suspicious document. Second, text alignment, where each candidate source document is compared to the suspicious document, extracting all passages of text that are highly similar. Third, knowledge-based post-processing, where the extracted passage pairs are cleaned, filtered and possibly visualized for later inspection [62].

3.1.1 Source Retrieval

Source retrieval is the initial phase in the process of plagiarism detection where, from a vast collection of documents, like an encyclopedia or the entire web, we constrain the collection to a smaller set of documents. Then we are going to take a closer look to decipher if plagiarism occurred by finding the exact passages that were plagiarized. Then, the source retrieval task turns into an information retrieval one where we are looking for potential sources of the suspicious document.

Concretely, PAN organizers define source retrieval as given a suspicious

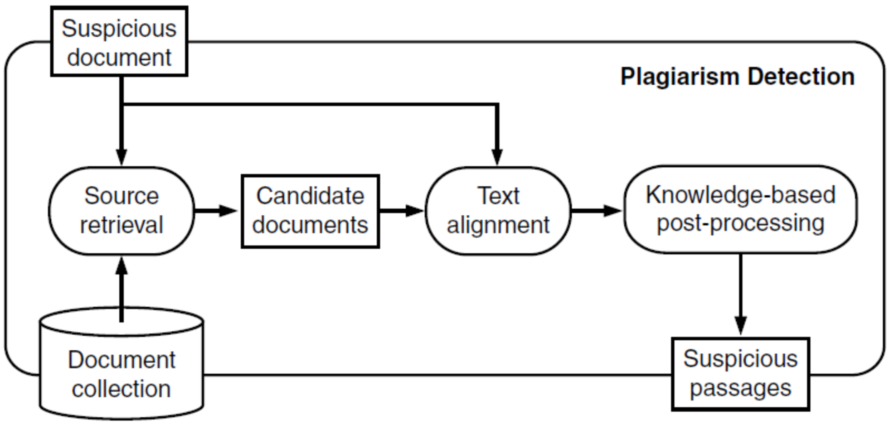


Figure 3.1: Generic retrieval process to detect plagiarism [81]

document and a web search engine, retrieve all source documents from which text has been reused while minimizing retrieval costs. The cost-effectiveness of plagiarism detectors in this task is important since using existing search engines is perhaps the only feasible way for researchers as well as small and medium-sized businesses to implement detection against the web, whereas search companies charge considerable fees for automatic usage [63].

3.1.2 Text Alignment

The text alignment task consists in identifying contiguous passages of reused text given a pair of documents. Most of the text alignment models follow a three-step approach: seeding, extension, and filtering [62]. The first step consists in finding relations (so-called “seeds”) between features extracted from the documents. At this stage, it is important to determine which type of features to extract and what kind of relation to search. For example, the features could be word n-grams with several implementations like Context n-grams [56,69,76,82], Context skip n-gram [69], Stop words n-grams [76,82] and Named Entity n-grams [76]. In our approach, we extracted sentences and compared them in a Vector Space Model (VSM) using the cosine similarity alike [32]. We also used the Dice coefficient as in [33] given that this measure favors an equal vocabulary distribution employed in the passages to be compared.

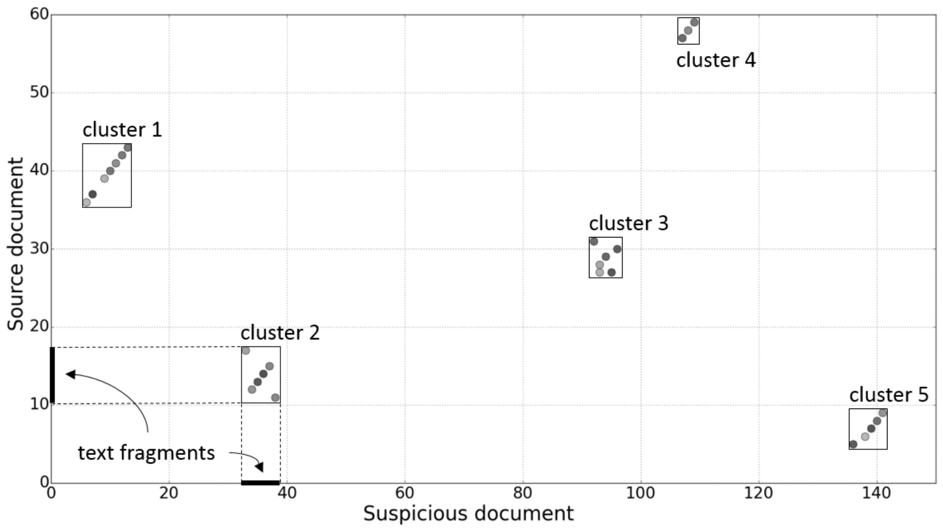


Figure 3.2: Clusters obtained after the extension step. The fragments of text (ranges of sentences) corresponding to cluster 2 are shown

Taking into account only the seeds extracted, some passages that do not show high similarity but are part of a plagiarism case could be missed. This behavior is due to the presence of noise, and also because a specific type of feature or similarity measure does not necessarily identify all possible types of obfuscation techniques.[1]

Accordingly, the extension step consists in joining these seeds into larger fragments. This is the core of a text alignment model. The basic idea here is to cluster together nearby seeds. A plagiarism case then is defined by the edges of a cluster: if we draw a rectangle around the cluster, the plagiarism case is the fragment of text in the suspicious document and its corresponding counterpart in the source document, as shown in Figure 3.2. Defining a cluster by its edges and not as a set of seeds, allows for small gaps in the range of seeds, which can be part of the plagiarism case even if the seeding process did not detect them; for example, see cluster 1 in the figure.

However, the greater the distance allowed between seeds in a cluster, the greater the probability of including passages that do not belong to the plagiarism case. Measuring the quality of a plagiarism case includes computing

[1]Obfuscation techniques refers to the changes done to the plagiarized passages like sentence reordering, changing words with synonyms, using summaries, among others.

the similarity between the two fragments of text. Thus, the challenge for an extension algorithm is to find a balance between the dispersion in clusters and the similarity of the fragments of text these clusters correlates. A problem that arises in the search for this balance in our approach is that the sentences do not necessarily have the same length, so a distance suitable for one cluster is not necessarily good for another cluster given the resulting similarity between the fragments of text. Therefore, balancing should be done for each cluster independently after the initial iteration of clustering is done.

Another significant problem when building an extension method is to determine what type of measure of distance should we use, and this is not a trivial problem. From the dots in Figure 3.2, it is expected to have clusters such as those represented, which relate a fragment of text in the suspicious document with a fragment of text in the source document. However, a Euclidean distance clustering algorithm as in [56] will fail to detect cluster 2, because two of its points are far from the rest of the group using this distance. These seeds in cluster 2 represent just a reordering of sentences: for instance, changing the last sentence in the source document to the first one in the suspicious document. Another way to compute distance could be using a function that returns the minimum distance in either dimension. This would result in accurate detection of cluster 2, but also would join clusters 2 and 5, because they are close by on the source document axis. Given that the two measures mentioned above compute the distance taking into account both dimensions at the same time, we used a method that computes the distance in one dimension at a time, alternating between them until no more division is needed. Several participants used algorithms in this direction taking into consideration the distance in characters [33, 69, 76, 82] or sentences [32].

The final step in the text alignment task is responsible for filtering out those clusters that do not meet certain criteria. Usually, this includes removing too short plagiarism cases or treating overlapping cases. The main problem we found using the PAN 2013 training corpus was that some plagiarism cases are contained inside larger cases in any of the two sides. To solve this problem, we introduced a measure of quality that compares overlapped cases, to decide which one to keep and which one to discard.

Finally, given that the three-step model for text alignment uses many parameters, it is impossible to find one optimal setting for all types of obfuscation. Therefore, the model should be adaptive; it should use heuristics to decide which type of obfuscation it is dealing with in a given document and choose the corresponding settings optimized for each type of obfuscation.

Table 3.1: Main ideas used in the systems participating in PAN 2012 and 2013

Stage	Method	[32]	[69]	[82]	[76]	[56]	[33]	Our
Preprocessing	Special character removal	+	−	−	−	−	−	+
	Number removal	−	−	−	−	+	−	−
	Stopword removal	+	+	−	−	−	−	−
	Case folding	+	+	+	+	+	−	+
	Stemming	+	+	−	−	+	−	+
Seeding	Bag of words	+	−	−	−	−	+	+
	Context n-grams	−	+	+	+	+	−	−
	Context skip n-grams	−	+	−	−	−	−	−
	Stop word n-grams	−	−	+	+	−	−	−
	Named entity n-grams	−	−	−	+	−	−	−
Extension	Bilateral Alternating Sorting	+	−	−	−	−	−	−
	Distance between seeds	+	+	+	+	−	+	+
	Euclidean distance clusters	−	−	−	−	+	−	−
	Multi-feature extension	−	+	−	+	−	−	−
Filtering	Passage similarity	+	−	−	−	−	−	+
	Small passage removal	−	+	+	−	+	−	+
	Overlapping removal	−	−	+	+	−	−	+
	Nearby passage joining	−	−	−	+	−	−	−

Table 3.1 summarizes the main ideas employed by the systems participating in PAN 2012 and 2013 [21, 32, 33, 56, 69, 76, 82], classified by the four main stages of a typical alignment process as suggested in [63].

3.1.3 PAN Shared Tasks

PAN is a conjunction of shared tasks that have been evolving since 2009 and that they divide into broad categories. In the 2013–2014 editions, they divided it into Authorship, Originality, and Trust. Since 2009, several tasks has surged and ended as part of their life cycle. Plagiarism detection has been an important field at PAN evolving with time. First, from 2009 to 2011 they organized the Intrinsic and External Plagiarism Detection tasks. Then, these tasks evolved to Source Retrieval and Text Alignment from 2012 to 2015. The scope of our work is the Text Alignment task from 2013 to 2015. In Figure 3.3 we highlight the task we proposed an approach that ranked

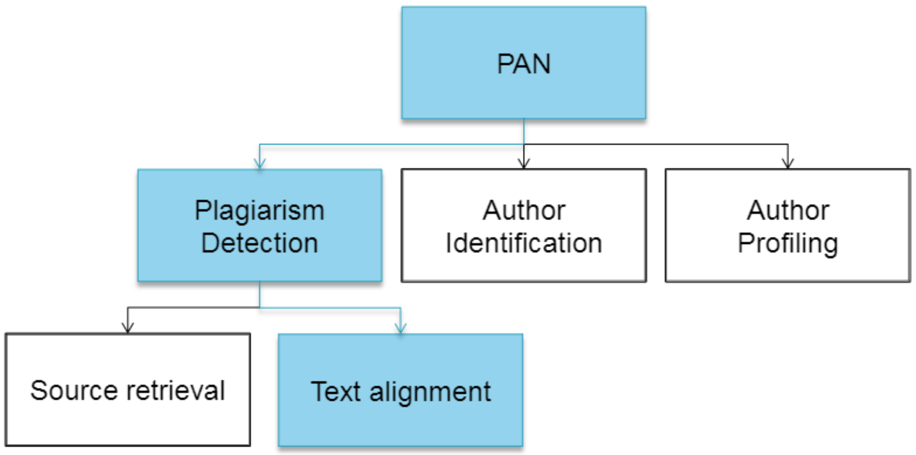


Figure 3.3: Our participation at PAN 2014

1st and that we describe in the section Our Approach to Text Alignment at PAN.

3.1.4 PAN Text Alignment Corpus

PAN workshops have developed several corpora for the text alignment task since its creation in 2009 [65]. These resources have evolved till the 2014 corpus where the life cycle of the text alignment task ended. The results we reported in the State-of-the-Art section were obtained by using this particular corpus. The generation process and structure of the corpus was detailed at [63]. Below, we summarize the main parts of this process.

They constructed the corpus in eight steps:

1. *Documents.* They used web documents obtained from the ClueWeb 2009 corpus distributed into 145 topics.

2. *Pre-Processing.* They converted the HTML documents to plain text, extracting the main text sections using the BoilerPipe model and adding some constraints regarding the number of words.

3. *Withheld Documents.* For each topic, they withheld one document in order no to be used for plagiarism.

4. *Source Set Formation.* Sets of source documents are generated by randomly selecting a topic and documents within this topic. Cosine similarity between the documents in a set is validated to be above a certain threshold to avoid unintended duplication between pairs of documents, which in turn may mislead text alignment algorithms.

5. *Withheld Source Sets.* For each topic, an additional source set was created but ensuring that no sentence of a source document has a duplicate sentence in the withheld document of that topic (chosen in Step 3). They consider a duplicate if the cosine similarity is above 0.9.

6. *Passage Extraction.* For a given source set created in Step 4, they extract passages ensuring that: there are not duplicate passages, there are not adjacent passages, and that there is at least a passage per document.

7. *Obfuscation.* Every passage from the previous step is obfuscated following four strategies: no obfuscation, random obfuscation, cyclic translation, and summary obfuscation. Below, we describe these strategies.

8. *Suspicious Document Generation.* A suspicious document is generated by randomly concatenating an (obfuscated) passage set.

   In the end, the corpus consists of pairs of suspicious and source documents in an XML format as shown in the following example:

```
<document reference="suspicious-document00006.txt">
<feature name="plagiarism"
    obfuscation="random"
    obfuscation_degree="0.4672857029362325"
    source_length="402"
    source_offset="1084"
    source_reference="source-document00560.txt"
    this_length="269"
    this_offset="1412"
    type="artificial"
/>
...
</document>
```

3.1.5 Evaluation Metrics

The lack of an evaluation metric drove the organizers of PAN in the context of their benchmarking workshop [61, 65] to propose the following metrics [64], which have been in use since 2009 and became the baseline in plagiarism detection.

Let d_plg denote a document that contains plagiarism. A plagiarism case in d_plg is a 4-tuple $s =< s_plg, d_plg, s_src, d_src >$, where s_plg is a plagiarized passage in d_plg, and s_src is its original counterpart in some source document d_src. Likewise, a plagiarism detection for document d_plg is denoted as $r =< r_plg, d_plg, r_src, d'_src >$; r associates an allegedly plagiarized passage r_plg in d_plg with a passage r_src in d'_src. It is said that $rdetectss$ iff $r_plg \cap s_plg \neg \phi$, $r_src \cap s_src \neg \phi$, and $d'_src = d_src$. With regard to a plagiarized document d_plg, it is assumed that different plagiarized passages of d_plg do not intersect; with regard to detections for d_plg, no such restriction applies. Finally, S and R denote sets of plagiarism cases and detections.

While the above 4-tuples resemble an intuitive view of plagiarism detection the authors resort to an equivalent, more concise view to simplify the subsequent notations: a document d is represented as a set of references to its characters $d = (1, d), ..., (|d|, d)$, where (i, d) refers to the $i - th$ character in d. A plagiarism case s can then be represented as $s = s_plg \cup s_src$, where $s_plg \subseteq d_plg$ and $s_src \subseteq d_src$. The characters referred to in s_plg and s_src form the passages s_plg and s_src. Likewise, a detection r can be represented as $r = r_plg \cup r_src$. It follows that $rdetectss$ iff $r_plg \cap s_plg \neg \phi$ and $r_src \cap s_src \neg \phi$. Based on these representations, the precision and recall of R under S are defined as follows:

$$prec\,(S, R) = \frac{1}{|R|} \sum_{r \in R} \frac{\left| \bigcup_{s \in S} (s \sqcap r) \right|}{|r|} \tag{3.1}$$

$$rec\,(S, R) = \frac{1}{|S|} \sum_{s \in S} \frac{\left| \bigcup_{r \in R} (s \sqcap r) \right|}{|s|} \tag{3.2}$$

where

$$s \sqcap r = \begin{cases} s \cap r & \text{if } r \text{ detects } s, \\ 0 & \text{otherwise.} \end{cases}$$

Besides precision and recall, there is another concept that characterizes the power of a detection algorithm, namely, whether a plagiarism case $s \in S$

is detected as a whole or in several pieces. The latter can be observed in today's commercial plagiarism detectors, and the user is left to combine these pieces to a consistent approximation of s. Ideally, an algorithm should report detections R in a one-to-one manner to the true cases S. To capture this characteristic we define the detection granularity of R under S:

$$gran\,(S, R) = \frac{1}{|S_R|} \sum_{s \in S_R} |R_s|\,, \tag{3.3}$$

where $S_R \subseteq S$ are cases detected by detections in R, and $R_s \subseteq R$ are the detections of a given s:

$$\begin{aligned} S_R &= \{s | s \in S \wedge \exists r \in R : r \text{ detects } s\}\,, \\ R_s &= \{r | r \in R \wedge r \text{ detects } s\}\,. \end{aligned}$$

The domain of $gran(S, R)$ is $[1, |R|]$, with 1 indicating the desired one-to-one correspondence and $|R|$ showing the worst case, where a single $s \in S$ is detected over and over again.

Precision, recall, and granularity allow for a partial ordering among plagiarism detection algorithms. To obtain an absolute order, they must be combined to define an overall score:

$$plagdet\,(S, R) = \frac{F_\alpha}{\log_2\left(1 + gran\,(S, R)\right)}, \tag{3.4}$$

where F_α denotes the F_α-Measure, i.e., the weighted harmonic mean of precision and recall. The authors suggest using $\alpha = 1$ (precision and recall equally weighted) since there is currently no indication that either of the two is more important. We take the logarithm of the granularity to decrease its impact on the overall score.

In Figure 3.4 is shown a document as a character sequence, either the suspicious or source document, that helps understand the rationale of the metrics proposed by Potthast et al.

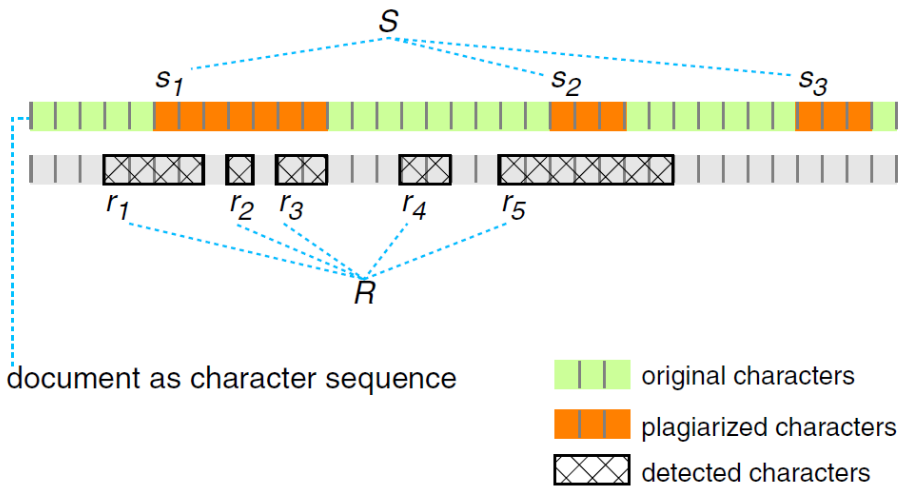


Figure 3.4: A document as character sequence, including plagiarized sections S and detections R returned by a plagiarism detection algorithm

3.2 Paraphrase Identification

According to Merriam-Webster dictionary, paraphrase is a restatement of a text, passage, or work giving the meaning in another form. Similarly, Bhagat et al. Bhagat and Hovy [8] define paraphrase as sentences or phrases that convey the same meaning using different wording.

Paraphrase identification is the task of determining whether two sentences have essentially the same meaning. This task has been shown to play a major role in many natural language applications, including text summarization, question answering, machine translation, natural language generation, and plagiarism detection. For example, detecting paraphrase sentences would help a text summarization system to avoid adding redundant information [2].

3.2.1 Typology

One reason why paraphrase recognition systems have been difficult to build is that paraphrases are hard to define. Although the strict interpretation of the term "paraphrase" is quite narrow because it requires exactly identical meaning, in linguistics literature paraphrases are most often characterized by an approximate equivalence of meaning across sentences or phrases [8].

To give some structure to the process of paraphrase recognition, several typologies have been proposed depending on the field of study. These typologies have significant differences in nature, whether they are focused on discourse analysis, linguistics, or computational linguistics. Cedeño et al. [5] classified the typologies created by several authors according to its field of study. In Table 3.2 we summarize their analysis.

From this analysis, Cedeño et al. [5] proposed the typology presented in Figure 3.5. Their paraphrase typology attempts to capture the general linguistic phenomena of paraphrasing, rather than presenting a long, fine-grained, and inevitably incomplete list of concrete mechanisms. In this sense, it also attempts to be comprehensive of paraphrasing as a whole. It was contrasted with, and sometimes inspired by state-of-the-art paraphrase typologies to cover the phenomena described in them; and it was used to annotate (i) the plagiarism paraphrases in the P4P corpus, (ii) 3,900 paraphrases from the news domain in the Microsoft Research Paraphrase Corpus (MSRP) [15], and (iii) 1,000 relational paraphrases (i.e., paraphrases expressing a relation between two entities) extracted from the Wikipedia-based Relational Para-

Table 3.2: Typologies classified according to their field of study

Field of study	Description	Authors
Discourse analysis	Reformulation mechanisms or communicative intention behind paraphrase	Gülich [24] & Cheung [12]
Linguistic analysis	Concrete theoretical frameworks, as the case of Meaning-Text Theory	Mel'čuk [46] & Milićević [50]
Linguistic analysis	Transformations and diathesis alternations focusing on on lexical and syntactic phenomena	Chomsky [13] & Harris [25] & Levin [37]
Linguistic-related fields	Editing	Faigley and Witte [16]

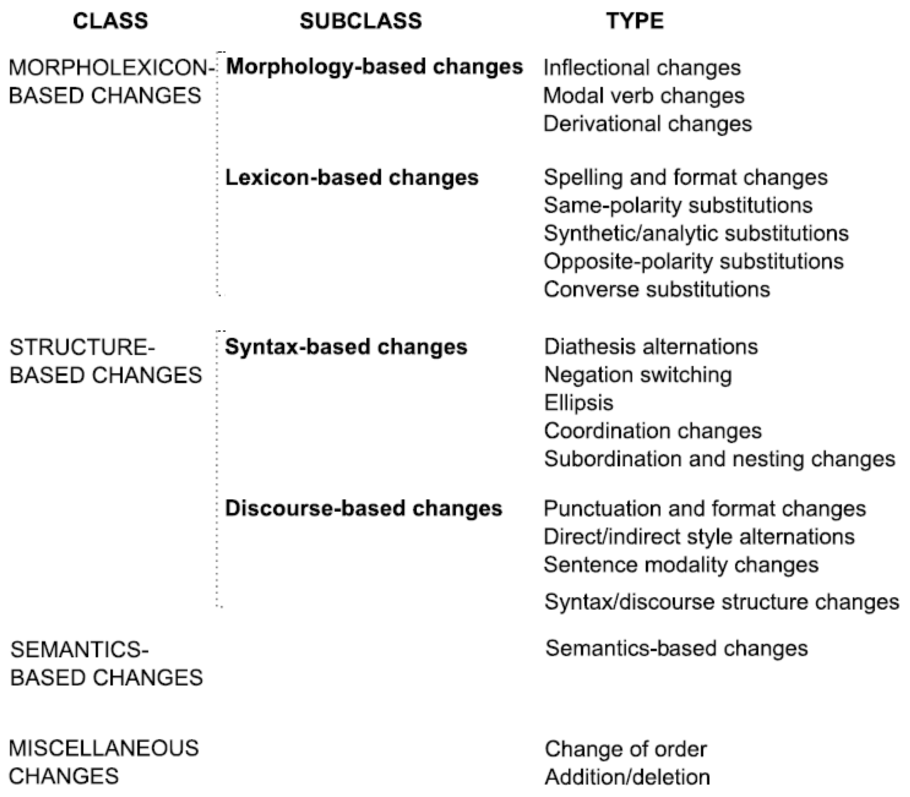
phrase Acquisition corpus (WRPA). P4P and MSRP are English corpora, whereas WRPA is a Spanish one. The success in the annotation of such diverse corpora with their paraphrase typology guarantees its adequacy for general paraphrasing, not only in English.

3.2.2 Corpora

P4P

Cedeño et al. introduced the P4P corpus in their work *Plagiarism Meets Paraphrasing: Insights for the Next Generation in Automatic Plagiarism Detection* [5]. The corpus was built using cases of simulated plagiarism in the PAN-PC-10 corpus. Here, simulated stands for those cases created by humans paraphrasing small documents, and were generated through the crowdsourcing Internet marketplace Amazon Mechanical Turk. The cases selected should contain 50 words or less, resulting in 847 paraphrase pairs meeting this condition.

After tokenization of the working corpus, the annotation was performed by, on the one hand, tagging the paraphrase phenomena present in each



CLASS	SUBCLASS	TYPE
MORPHOLEXICON-BASED CHANGES	**Morphology-based changes**	Inflectional changes
		Modal verb changes
		Derivational changes
	Lexicon-based changes	Spelling and format changes
		Same-polarity substitutions
		Synthetic/analytic substitutions
		Opposite-polarity substitutions
		Converse substitutions
STRUCTURE-BASED CHANGES	**Syntax-based changes**	Diathesis alternations
		Negation switching
		Ellipsis
		Coordination changes
		Subordination and nesting changes
	Discourse-based changes	Punctuation and format changes
		Direct/indirect style alternations
		Sentence modality changes
		Syntax/discourse structure changes
SEMANTICS-BASED CHANGES		Semantics-based changes
MISCELLANEOUS CHANGES		Change of order
		Addition/deletion

Figure 3.5: Paraphrase typology

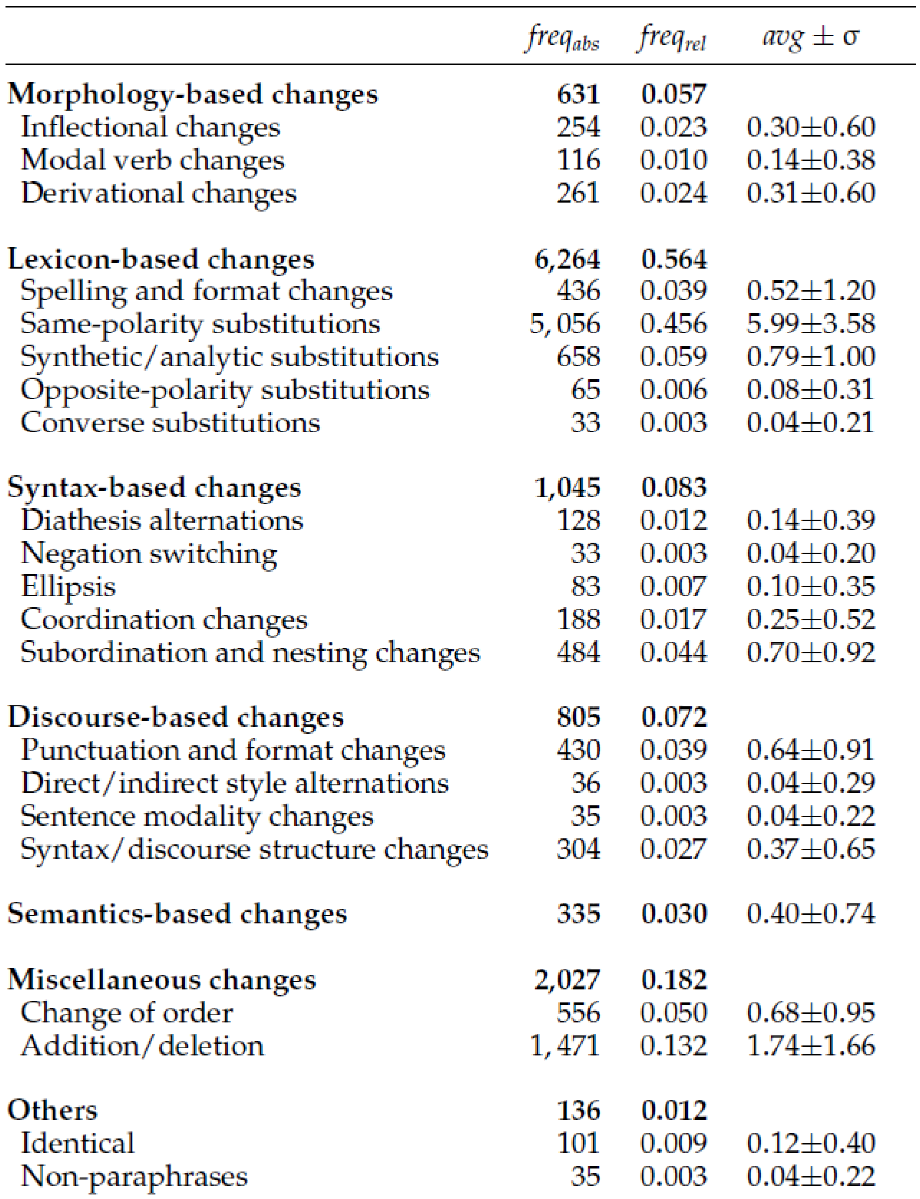
source/plagiarism pair with their tag set (each pair contains multiple paraphrase phenomena). The tag set the authors proposed is described in the Section 2.3.1 (Typology) and shown in Figure 3.5. On the other hand, indicating the scope of each of these tags (the range of the fragment affected by each paraphrase phenomenon). Their tag set consists of 20 paraphrase types plus identical and non-paraphrase tags. Identical refers to those text fragments of the source/plagiarism pairs that are exact copies; non-paraphrase refers to fragments of the source/target pairs that are not semantically related. The reason for adding these two tags was to see how they perform in comparison to the actual paraphrase cases [5].

Regarding the scope, they do not annotate strings but linguistic units (words, phrases, clauses, and sentences). The scope affects the annotation task differently regarding the classes [5].

For the classes *morpholexicon-based changes, semantics-based changes, and miscellaneous changes*, only the linguistic unit(s) affected by the trigger change is (are) tagged. As some of these changes entail other changes, two different attributes were provided. LOCAL, which stands for those cases in which the trigger change does not entail any other change in the sentence; and GLOBAL, which means those cases in which the trigger change does entail other changes in the sentence [5].

For the class *structure-based changes*, the whole linguistic unit suffering the syntactic or discourse reorganization is tagged. Moreover, most structure-based changes have a key element that gives rise to the change and distinguishes it from others. This key element is also tagged [5].

Paraphrase type frequencies and total and average lengths are shown in Figure 3.6. Same-polarity substitutions represent the most frequent paraphrase type. At a considerable distance, the second most common type is addition/deletion. The authors suppose that the way paraphrases were collected in PAN-PC-10 corpus had a major impact on these results. They were created manually, asking people to simulate plagiarizing by re-writing a collection of text fragments–that is, they were originated in a reformulation framework, where a conscious reformulating intention by a speaker exists. Their hypothesis is that the most frequent paraphrase types in the P4P corpus correspond to the paraphrase mechanisms most accessible to humans when asked to reformulate or plagiarize. Same-polarity substitutions and addition/deletion are mechanisms that are relatively simple to apply to a text by humans: changing one lexical unit for its synonym (understanding synonymy in a general sense) and deleting a text fragment, respectively [5].



	$freq_{abs}$	$freq_{rel}$	$avg \pm \sigma$
Morphology-based changes	**631**	**0.057**	
Inflectional changes	254	0.023	0.30±0.60
Modal verb changes	116	0.010	0.14±0.38
Derivational changes	261	0.024	0.31±0.60
Lexicon-based changes	**6,264**	**0.564**	
Spelling and format changes	436	0.039	0.52±1.20
Same-polarity substitutions	5,056	0.456	5.99±3.58
Synthetic/analytic substitutions	658	0.059	0.79±1.00
Opposite-polarity substitutions	65	0.006	0.08±0.31
Converse substitutions	33	0.003	0.04±0.21
Syntax-based changes	**1,045**	**0.083**	
Diathesis alternations	128	0.012	0.14±0.39
Negation switching	33	0.003	0.04±0.20
Ellipsis	83	0.007	0.10±0.35
Coordination changes	188	0.017	0.25±0.52
Subordination and nesting changes	484	0.044	0.70±0.92
Discourse-based changes	**805**	**0.072**	
Punctuation and format changes	430	0.039	0.64±0.91
Direct/indirect style alternations	36	0.003	0.04±0.29
Sentence modality changes	35	0.003	0.04±0.22
Syntax/discourse structure changes	304	0.027	0.37±0.65
Semantics-based changes	**335**	**0.030**	0.40±0.74
Miscellaneous changes	**2,027**	**0.182**	
Change of order	556	0.050	0.68±0.95
Addition/deletion	1,471	0.132	1.74±1.66
Others	**136**	**0.012**	
Identical	101	0.009	0.12±0.40
Non-paraphrases	35	0.003	0.04±0.22

Figure 3.6: Absolute and relative frequencies of the paraphrase phenomena occurring within the 847 source–plagiarism pairs in the P4P corpus

Table 3.3: MSRP distribution

	positives	negatives	total
train	2753	1323	4076
test	1147	578	1725
total	3900	1901	5801

Microsoft Research Paraphrase Corpus

The Microsoft Research Paraphrase Corpus (MSRP) is widely used in the Paraphrase Recognition/Identification task, being the baseline to compare different algorithms. The corpus contains 5801 pairs of sentences which have been extracted from news sources on the web, along with human annotations indicating whether each pair captures a paraphrase/semantic equivalence relationship [66] [15]. An example pair tagged as a paraphrase or "semantically equivalent" is:

- Amrozi accused his brother, whom he called "the witness", of deliberately distorting his evidence.
- Referring to him as only "the witness", Amrozi accused his brother of deliberately distorting his evidence.

The corpus is divided into training and test dataset with a certain distribution of positive (paraphrase) and negative (non-paraphrase) pairs, as shown in Table 3.3.

The authors, interested in identifying more complex paraphrase relationships, restricted the dataset to a minimum word-based Levenshtein distance of 8. This constraint helped avoid the most trivial sorts of paraphrase, such as sentence pairs differing only a single word.

Each pair of sentences was examined by two human annotators who were asked to give a binary judgment as to whether the two sentences could be considered "semantically equivalent", while a 3rd judge was used to resolve disagreements. The methodology followed for the annotation process was: Rater 1 scored all 5801 sentences. Rater 2 scored 3533 sentences, and Rater 3 scored 2268 sentences. For the sentences where Rater 1 and 2 did not agree with the judgment, Rater 3 gave a final judgment, while Rater 2 gave the final judgment on sentences where Rater 1 and Rater 3 did not agree. The inter-rater agreement is shown in Table 3.4.

Table 3.4: MSRP interrater agreement

	Scored	Agreements	Percentage
Raters 1 & 2	3533	2904	82.20%
Raters 1 & 3	2268	1921	84.70%

Paraphrase Database

The first version (1.0) of the Paraphrase Database (PPDB) [20] contained two large datasets containing millions of paraphrases for the English and Spanish language. In the following version (2.0) [19], they included several other languages. The database was automatically extracted following Bannard and Callison-Burch [3] intuition that two English strings that translate to the same foreign string can be assumed to have the same meaning. For this purpose, they used several parallel corpora. We focus on the English dataset of last version [58] which is divided in different sizes and options. There are three options:

- Lexical: single word to single word
- Phrasal: multiword to single/multiword
- Syntactic: paraphrase rules containing non-terminal symbols

There are also different available sizes (S, M, L, XL, XXL, XXXL) ranging from 550MG to 17 GB. There are other considerations and combinations that can be downloaded from the authors site[2].

The corpus is distributed as plain text files where each line is one paraphrase in the format:

LHS ||| PHRASE ||| PARAPHRASE ||| (FEATURE=VALUE)* ||| ALIGNMENT ||| ENTAILMENT,

where LHS is the constituent and FEATURES is a list with values such as estimated paraphrase and entailment probabilities.

[2]algo, 14/05/2018

3.2.3 Distributional Approaches

Most recent approaches are leaning towards distributional models. Most of these models obey a well defined general structure. First, a distributional word representation (word embeddings) followed by a compositional model to generate phrase representations, and finally, a similarity layer to compare both phrase representations. There are several adaptations inside these general steps regarding the way they interact with each other. In some approaches the models are trained end-to-end over the same dataset from computing the embeddings, the compositional model and similarity score between the pair of sentences. Other variations include using word embeddings trained on large datasets, training the compositional model in a different and bigger dataset to our particular task, jointly training the embeddings and compositional model, among others. The models trained on different datasets aim to apply transfer learning to the task of paraphrase identification and just fine-tune the previously learned parameters.

3.2.4 Word Representation

Word representation, specifically distributional models, assume that words that occur in similar contexts have similar meanings. The aim then is to construct a word space where these words are represented as similar vectors. Some approaches rely on co-occurrence count matrices using a given corpus. Recent approaches however, show a shift towards neural network models were the word space is optimized in a prediction task. These models aim to maximize the probability of observing a word in a given context or vice-versa. We can also find combinations of both of these models.

Mitchell & Lapata

Mitchell & Lapata [52] proposed a simple distributional semantic space. They represent target words in an M-dimensional space where M are the most frequent non-stopword terms in the British National Corpus (BNC). Then, they compute co-occurrence counts within a context window in either direction of the target words. Equations (3.5) and (3.6), extracted from Blacoe & Lapata analysis [9], detail this method. The vectors dimensions will be given by

$ctxt_{top} = \left\{ w_1^{(top)}, ..., w_M^{(top)} \right\}$ and the co-ocurrence count will be:

$$coCount_w[j] = \sum_{i=1}^{n_{BNC}} \sum_{t=1}^{n_i} \left| \left\{ k \in [t-5; t+5] | w_t^{(i)} = w, w_k^{(i)} = w_j^{(top)} \right\} \right|, \quad (3.5)$$

for $w \in Voc_{BNC}$ and $j = 1, ..., M$. Then, each word representation is given by:

$$wdVec_w^{(rp)}[j] = \frac{p(w_j^{(top)}|w)}{p(w_j^{(top)})} = \frac{coCount_w[j]}{freq_w} \times \frac{totalCount}{freq_{w_j^{(top)}}}, \quad (3.6)$$

for $j = 1, ..., M$, where *totalCount* is the total number of words in the BNC.

In recent years, several authors have embraced the use of neural language models to address the task of paraphrase identification. Some of the most populars being the language model proposed by Collobert & Weston [14], the skip-gram & CBOW models (word2vec) by Mikolov et al. [49], and the GloVe model by Pennington et al. [59].

Collobert & Weston

Collobert & Weston convert the unsupervised learning task of computing word embeddings to a binary classification task. Given a fixed-size window and the middle word in as the positive samples they generate the negative samples by replacing the middle word at random. This way they enforce semantic coherence. The objective function of the model is a ranking-type cost:

$$\sum_{s \in S} \sum_{w \in D} \max(0, 1 - f(s) + f(s^w)), \quad (3.7)$$

where S is the set of windows, D is the dictionary of words, and $f(\cdot)$ represents their Neural Network (NN) model consisting of a convolution, max Pooling, and fully-connected NN layers, where the last layer returns a single output.

One implementation available in GitHub[3] concatenates all the embeddings of the fixed-size window and use a two fully-connected layers NN. In this implementation the only generate a 100 negative samples per positive one and not using the entire vocabulary which would require a lot of processing.

[3]`https://github.com/klb3713/cw_word_embedding`

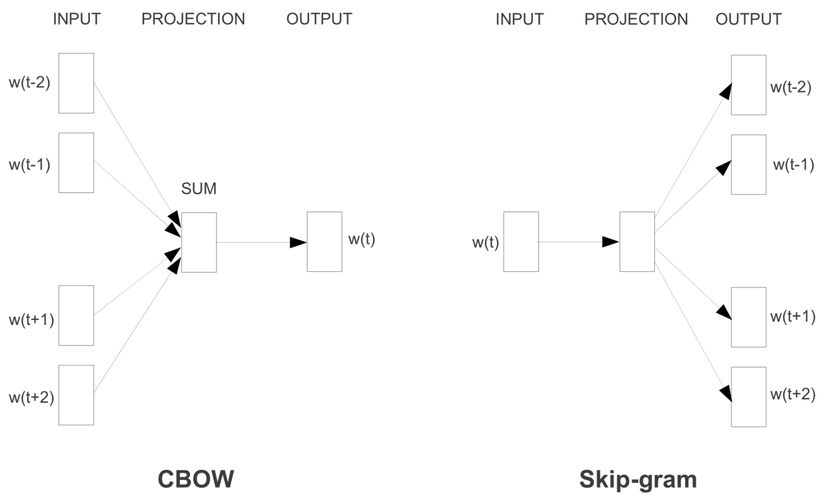


Figure 3.7: CBOW and Skip-gram models [48]

Word2vec

Mikolov et al. [48] proposed two architectures as part of their word2vec model. Both are based on the neural probabilistic model proposed by Bengio et al. [6], which consist of input (one-hot word representations), projection (vectorial word representations), hidden and output layers.

The first of their architectures is the Continuous Bag of Words (CBOW) where they propose a log-linear classifier to predict the middle word given a set of context words. They remove the hidden layer from the neural language model and average the word vectors of the given context after the projection layer. Since the use average the order of the context words does not matter, hence the name bag of words (BOW). Continuous BOW comes from the fact they use continuous distributed representation of the context.

The second architecture is the Continuous Skip-gram (Skip-gram). They maximize the classification of a word based on another word in the context. The target word is the input of the log-linear classifier and predicts words from the context.

Figure 3.7, extracted from their article, provides a better understanding of the models rationale.

The authors have performed subsequent experiments and made further improvements that can be found in their follow up work [49]. Additionally, there is a popular implementation of these models in the gensim Python module [67]. There are also some pre-trained word vectors freely available in the Internet[4].

Global Vectors (GloVe)

Global Vectors (GloVe) was proposed by Pennington et al. [59] which is a method based on word-word co-occurrence counts extracted from a given corpus. The idea is to count the number of times a word j occurs in the context of word i. This allows to compute a probability given by the co-occurrence count of j and i divided by the frequency of target word j. They compute probability ratios of two target words to a set of probe words k instead of just single probabilities. They introduce a weighted least squares regression model that takes into account the frequency of co-occurrences.

There is an official release implementation of this approach together with some pre-trained word embeddings using different dimensions and corpora[5].

Cheng & Kartsaklis

Cheng & Kartsaklis [11] compute the embeddings jointly with the compositional model. They extend Collobert & Weston's model (CW) to be aware of syntax. To achieve this goal, beside generating a set of negative examples where the middle word in the fixed-size window is replaced by a random word (retaining semantic), they also generate a second set of negative examples where the context of the middle word have been randomly shuffled (retaining syntax). They replace the NN model used in CW's with a recurrent neural network (RNN) or a recursive neural network (RecNN), and a fully-connected layer to their output. They also extend their model to disambiguate the embeddings. They follow the multi-sense model of Neelakantan et al. [54] assigning a fixed number n of senses to each word and using a gated architecture to select one of them.

Others

When creating a NN-based approach for paraphrase identification or similar tasks, which requires a compositional model, the embedding layer is

[4]urlhttps://code.google.com/archive/p/word2vec/

[5]`https://nlp.stanford.edu/projects/glove/`

implemented as a lookup table, usually, initialized with pre-trained embeddings and with the option of further tune them on the specific task. Several approaches use this technique. Yin & Schütze [87] initialize its embeddings with those provided by Turian et al. [83] (based on Collobert & Weston [14]) and then further tune them. He et al. [26] concatenates three embeddings: pre-trained 300-dim GloVe [59], 25-dim PARAGRAM [85] which were trained using the Paraphrase Database (PPDB) [20], and a 200-dim word2vec model [49] trained with (word, POS tags) on Xinhua News Agency corpus. Wang et al. [84] and Shen et al. [75] use the pre-trained word2vec embeddings provided by Mikolov [49]. Socher et al. [79] use Collobert and Weston's embeddings [14].

3.2.5 Compositional Model

Compositional models aim to compute phrase representations from individual words. There is a wide range of approaches for compositionality applied to paraphrase identification, from simple element-wise vector operators such as addition and multiplication [9] to complex deep learning models [11, 26, 75].

Ji & Eisenstein

Ji & Eisenstein [28] do not use distributional representations of words. Instead, they start by computing the Bag of Words (BOW) representation of the sentences using term-frequency as the weighting scheme. Then, they reweight each word in the vocabulary by computing its probability of appearing in both sentences of the pair and being a paraphrase sample. They use two Bernoulli (Equations (3.8)(3.9)) distributions and the Kullback-Leiber divergence (Equation (3.10)) to perform the reweighting. They update the BOW values with these probabilities. Finally, they compute a dimensionally-reduced latent representation of the sentences using nonnegative matrix factorization.

$$p_k = P(w_{ik}^{(1)} | w_{ik}^{(2)} = 1, r_i = 1), \tag{3.8}$$

is the probability that sentence $w_i^{(1)}$ contains feature k, given that k appears in $w_i^{(2)}$ and the two sentences are labeled as paraphrases, $r_i = 1$.

$$q_k = P(w_{ik}^{(1)} | w_{ik}^{(2)} = 1, r_i = 0), \tag{3.9}$$

is the probability that sentence $w_i^{(1)}$ contains feature k, given that k appears in $w_i^{(2)}$ and the two sentences are labeled as not paraphrases, $r_i = 0$.

$$KL(p_k||q_k) = \sum_x p_k(x) log \frac{p_k(x)}{q_k(x)}, \tag{3.10}$$

is the Kullback-Leibler divergence which measures the discriminability of feature k.

Cheng & Kartsaklis

As mentioned in previous section, Cheng & Kartsaklis [11] train the word embeddings and the compositional model in the same step in an unsupervised way. The use a recurrent or recursive neural network (they experiment with both architectures) with a fully connected layer after each time step to score the linguistic plausibility of the computed phrase representations.

Yoon Kim

Yoon Kim [31] propose a model based on Convolutional Neural Networks for sentence classification where he applies a variety of convolutional filters using different kernel sizes in parallel, followed by a Max pooling layer, and a fully connected layer applied to the concatenation of the pooling outputs. The kernel sizes are given by the n-grams being extracted and the dimensions of the embeddings (eg. $(1, 300), (2, 300) and (3, 300))$, which can be viewed as a "temporal" convolution as it convolve over regions of the phrase.

He et al.

He et al. [26] propose a combination of convolution filters and pooling methods to extract features from a sentence. Specifically, they proposed two types of filters, one based on Kim's work which they call holistic filters. The other, based on applying independent convolutions for each dimension of the embeddings which the call per-dimension filters. They combine the convolution operations with max, min, and mean pooling layers independently, i.e. a convolution for each pooling operation.

Wang et al.

Wang et al. [84] starts by representing each sequence of words in a sequence of embeddings using the pre-trained word2vec values. Then, given a sentence pair, they decompose each word representation into a two-components vector, "similar" s^+ and "dissimilar" s^- components. After this, they apply a two-channel CNN operation to compose both components into a feature vector. Unlike previous authors they compute the sentence representations by taking into account both sentences from the pair and hence their model is dependent on the specific task of paraphrase identification. To perform the decomposition of a given word s_i, they compute the cosine similarity $a_{i,j}$ between the given word and all words of the other sentence $T = \{t_0, ..., t_m\}$. Then, select the word that returns the highest value together with a context given a fixed-size window. They compute a semantic matching vector $\hat{s_i}$ for s_i in the following way:

$$\hat{s_i} = \frac{\sum\limits_{j=k-w}^{k+w} a_{i,j} t_j}{\sum\limits_{j=k-w}^{k+w} a_{i,j}}, \tag{3.11}$$

where k is the position of the most similar term between of T to s_i, $a_{i,j}$ is the cosine similarity, and w is the size of the window.

Given the word representation s_i and computed the semantic matching vector $\hat{s_i}$, the authors experimented with three decomposition functions. For simplicity we describe only the two best-performing functions.

Linear: This decomposition function assigns a bigger proportion of the original vector if the similarity with its semantic matching vector is higher. It is defined as:

$$\alpha = \cos(s_i, \hat{s_i}) \tag{3.12}$$

$$s_i^+ = \alpha s_i \tag{3.13}$$

$$s_i^- = (1-\alpha) s_i \tag{3.14}$$

Orthogonal: Decomposes a vector in the geometric space. It is defined as:

$$\begin{cases} s_i^+ = \frac{s_i \cdot \hat{s_i}}{\hat{s_i} \cdot \hat{s_i}} \hat{s_i} & \text{parallel} \\ s_i^- = s_i - s_i^+ & \text{perpendicular} \end{cases} \tag{3.15}$$

Socher et al.

Socher et al. [79] use a recursive autoencoder (RAE) applied to a given parse tree. The parent representations of the tree are computed from the children using a standard neural network layer. To asses the quality of these vectors representations they use another matrix to decode their vectors in a reconstruction layer and compute the Euclidean distance between the original input and its reconstruction. They also use an extended version called unfolding recursive autoencoder where the try to reconstruct all intermediate nodes of the tree.

Shen et al.

Shen et al. [75] uses a bidirectional LSTM with outputs in each time step to compute the representations of each sentence. The bidirectional LSTM consist on applying the recurrent network in both directions of the sentence (left-to-right and vice versa) and concatenating both outputs.

Yin & Schütze

Yin & Schütze [87] proposed an architecture based on convolutional, averaging and max-pooling layers following the model of Kalchbrenner et al. [30]. The procedure goes as: wide convolution, averaging, dynamic k-max pooling, wide convolution, averaging and k-max pooling. The rational of their model is obtain a multigranular sentence representation at four different levels: word (word embeddings), short ngram (first averaging layer), long ngram (second averaging layer), and sentence (output of last k-max pooling layer). Given a sequence of tokens S_i, a wide convolution consist in convolve a weight matrix $\mathrm{M} \in \mathbb{R}^{d \times m}$ over the sequence representation $S_i \in \mathbb{R}^{d \times |S_i|}$ generating a matrix $\mathrm{C} \in \mathbb{R}^{d \times (|S_i|+m-1)}$. Note that in this type of convolution the sequence need to be padded to achieve the desire dimensions in the resulting matrix. Also, they do not add any bias term of activation function. The averaging layer consist in computing the mean between odd and even rows of the embeddings, add the bias weights and apply tanh activation function. The averaging layer generates a matrix $A \in \mathbb{R}^{\frac{d}{2} \times (|S_i|+m-1)}$. The k-max pooling is a generalization of the max-pooling over time operation. It returns the top-k values in the temporal dimension. This layer ensures a fixed-size output. The dynamic k-max pooling returns the top-k_{dy} values where k_{dy} depends on the length of the sequence S_i.

$$k_{\text{dy}} = \max(k_{\text{top}}, |S_i|/2 + 1) \tag{3.16}$$

Blacoe & Lapata

Blacoe & Lapata [9] experimented with simple compositional models consisting of addition and multiplication of the word representations in a sequence.

3.2.6 Similarity Layer

The similarity layer compares the sentences representations in a way it can be feed to a classifier to resolve a particular task. Given a pair of sentences the paraphrase identification task consist in assign a value of 1 if both sentences convey the same meaning.

Ji & Eisenstein [28] compute a sample vector by concatenating the element-wise sum and absolute difference of both sentences. In addition to the sample vector, they use a set of hand-engineered features to capture fine-grained similarity between sentences.

Cheng & Kartsaklis [11] use a Siamese architecture where the compositional model weights are shared to compute the sentences representations. Given the sentence vectors $f(s_1)$ and $f(s_2)$, they experimented with two cost functions, the L_2 norm and the cosine similarity. The L_2 norm is defined as:

$$E_f = \begin{cases} \frac{1}{2}\left\|f(s_1) - f(s_2)\right\|_2^2, & \text{if} y = 1 \\ \frac{1}{2}\max(0, m - \left\|f(s_1) - f(s_2)\right\|_2)^2, & o.w. \end{cases} \tag{3.17}$$

where m stands for the margin hyper-parameter chosen in advance. The cosine similarity cost is defined as:

$$E_f = \frac{1}{2}(y - \sigma(wd + b))^2, \tag{3.18}$$

where d is the cosine similarity between both sentences, w and b are the scaling and shifting parameters to be optimized, σ is the sigmoid function.

Keeping in mind that their compositional model returns features for the combinations of holistic filters with max, min, mean pooling functions; and per-dimension filters with max and min pooling functions, He et al. [26] proposed two algorithms that compare features from certain “regions”. Both algorithms compared features from the same pooling function only. The first algorithm is applied only over the holistic filters. They compute two measures: the cosine similarity and the euclidean distance. The operations are applied over the concatenation of all convolutional kernel sizes and for each filter independently. The second algorithm can be divide in two parts. In the first part all kernel sizes of the holistic filters between the sentences are compared, in the second part for each kernel size and each filter of the per-dimension filters are compared. For this algorithm they compute three measures: the cosine similarity, the euclidean distance and the sum of absolute distance. After applying both algorithms they concatenate all the features and feed them to a two-layer traditional neural network with a softmax layer as final output to compute the similarity score of both sentences. See the authors paper for a detail description of their methods.

Wang et al. [84] concatenates the two sentence representations and apply a linear function with a sigmoid in the output to constrain the similarity within the range $[0,1]$.

In the work of Socher et al. [79], given the representation of each word and phrases for the nodes of the parse tree, they compute the Euclidean distances between all word and phrase vectors of the two sentences generating a similarity matrix S. Considering the sentence lengths as n and m, and since the parse tree is binary then $S \in \mathbb{R}^{(2n-1)\times(2m-1)}$. They propose a dynamic pooling method to transform S into $S_{pooled} \in \mathbb{R}^{n_p \times n_p}$. Each value of S_{pooled} is the minimum of each rectangular region formed by splitting the original matrix S into a grid of $\frac{2n-1}{n_p}$ by $\frac{2m-1}{n_p}$ regions. The feed the resulting fixed-size matrix to a standard softmax classifier.

Given their compositional architecture, Yin & Schütze [87] extract feature matrices by computing similarity between the two sentences on multiple levels, specifically what they call: unigram, short ngram, long ngram and sentence. They use the euclidean distance to compute the similarity defined as:

$$\hat{F}_l^{ij} = \exp\left(\frac{-||S_{:,i}^{1,l} - S_{:,j}^{2,l}||^2}{2\beta}\right), \tag{3.19}$$

where $-||S_{:,i}^{1,l} - S_{:,j}^{2,l}||^2$ is the Euclidean distance between the representation of the ith item of S_1 and jth item of S_2 on level l. They set $\beta = 2$.

As the matrices dimensions vary according to the sentence lengths they propose a dynamic pooling for feature matrices based on the method of Socher et al. [79]. Then, all fixed-size resulting matrices are concatenated and fed to a logistic regression classifier.

For comparing the two sentence representations, Shen et al. [75] propose an architecture which they name Gated Relevance Network (GRN). They incorporate a bilinear model and a single layer network through a gate mechanism. Given that their compositional model is based on a bidirectional LSTM that returns a representation for each time step, the bilinear model is defined as:

$$s(h_{x_i}, h_{y_i}) = h_{x_i}^T M h_{y_i}, \tag{3.20}$$

where $M \in \mathbb{R}^{d \times d}$ is a matrix to be learned and d is the dimension of the output of each time step of the LSTM. The single layer network is applied on the concatenation of the representations as:

$$s(h_{x_i}, h_{y_i}) = u^T f \left(W \begin{bmatrix} h_{x_i} \\ h_{y_j} \end{bmatrix} + b \right), \tag{3.21}$$

where f is the non-linear activation function, $W \in \mathbb{R}^{k \times 2d}$ and $b \in \mathbb{R}^k$ are the trainable weights. Then the definition of the GRN is:

$$s(h_{x_i}, h_{y_i}) = u^T \left(g \odot h_{x_i}^T M^{[1:r]} h_{y_i} + (1-g) \odot f \left(W \begin{bmatrix} h_{x_i} \\ h_{y_j} \end{bmatrix} + b \right) \right) \tag{3.22}$$

where the gate g is defined as:

$$g = \sigma \left(W_g \begin{bmatrix} h_{x_i} \\ h_{y_j} \end{bmatrix} + b_g \right), \tag{3.23}$$

The output of the GRN is a matrix with dimensions determined by the input sequence lengths. The next step they take is applying a max-pooling layer of non-overlapping regions. Given that they pad all the sequences of the dataset to a max_length size, applying GRN and max-pooling operations result in a fixed-size output. The next step is feeding that output to a fully-connected layer.

Blacoe & Lapata [9], experimented with some combinations of the sentence representations plus some hand-engineered features. These features are fed to a liblinear classifier. The features are: either the concatenation or subtraction of both vectors, a vector encoding which words appear in each sentence (sparse vector where the dimensions are the vocabulary of MSRP corpus), the cosine similarity, the length of both sentences, and the unigram overlap.

Chapter 4

Text Alignment

4.1 Text Alignment Improvements

4.1.1 Adaptive Behavior

At the PAN competition, the methods are evaluated on a corpus that contains plagiarism cases created using four different types of obfuscation: none, random, translation, and summary. In the training dataset, we have the option to test our approaches in sub-corpus divided according to the type of obfuscation used to create the plagiarism cases. We observed that the optimal parameters of our method are different to detect such diverse types of obfuscated plagiarism cases. Therefore, we introduced three alternative paths and decided which output to use according to the type of obfuscation we are likely dealing with in each specific document pair.

The final set up of our approach is shown in Figure 4.1. After initial preprocessing and seeding steps, we follow two separate paths with different *maxgap* values: one value (*maxgap_summary*) that we found to be best for the summary obfuscation sub-corpus and one that was best for the other three corpora (*maxgap*). After we obtain the plagiarism cases using these two different settings, we applied a verbatim detector method to the non-summary approach resulting in three possible results. We named these results as verbatim plagiarism cases (cases V), summary plagiarism cases (cases S), and regular plagiarism cases (cases O).

Verbatim Detector

The verbatim detection method is based on the Longest Common Substring (LCS) algorithm. We modify the LCS algorithm to use words instead of characters and to find every single common sequence of words above a certain threshold measured in characters (*th_verbatim*).

Output Selector

The decision of which of the three outputs (cases V, cases S and cases O) report as the final result of our approach follows a decision cascade where the verbatim plagiarism cases have priority, them summary and finally regular cases. All three possible outputs are mutually exclusive.

If there is at least one verbatim case, the pair of documents is considered as a non-obfuscated pair, and the verbatim output is reported. If none ver-

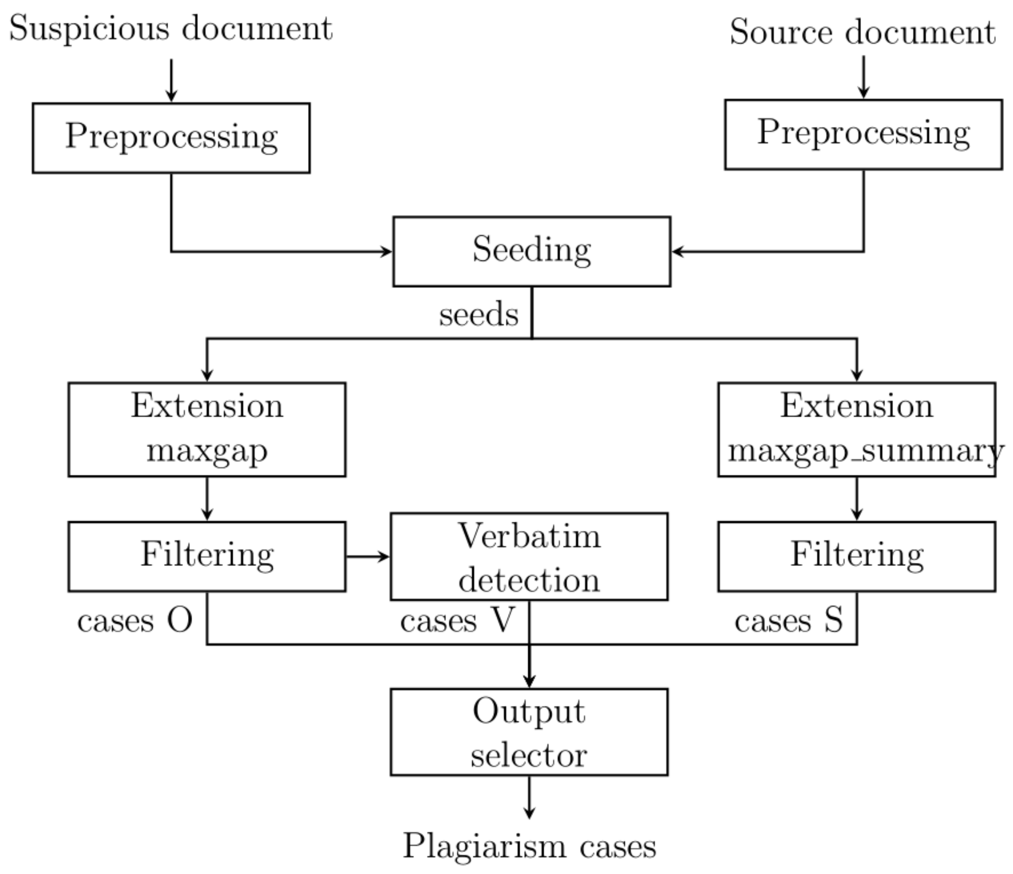


Figure 4.1: Diagram of the final approach

Table 4.1: Parameters settings

Parameter	Value	Parameter	Value
minsentlen	3	maxgap_summary	24
th_cos	0.30	maxgap_least	0
th_dice	0.33	minsize	1
th_validation	0.34	minplaglen	150
maxgap	4	th_verbatim	256

batim cases were reported, we decide whether cases S are likely to represent summary obfuscation or not. To judge this, we compare the relative length of the suggested suspicious fragments to the source fragments. Specifically, the decision is made based on the variables src_{len} and $susp_{len}$, which correspond to the total length of all passages, in characters, in the source document and the suspicious document, respectively. When $susp_{len}$ is much smaller than src_{len}, then we are likely dealing with summary obfuscation. If both, verbatim and summary cases where discarded then the reported output is regular plagiarism cases.

In table Table 4.1 we show the final setting of the parameters used in our system.

4.1.2 Results

We trained our system using the corpus provided for PAN 2014 competition and using the performance measures introduced in [64]. We compared this approach to our previous one in all of the sub-corpus and measurements. We tested our system in the PAN 2014 training corpus using the TIRA platform [23]. As observed in Table 4.2 the performance in the none obfuscated sub-corpus was increased dramatically because of the new verbatim detector. We also observed an increased in the summary sub-corpus because of the new parameters settings. These improvements came without affecting significantly the other sub-corpus and hence, the final result was increased as well.

The results showed that recall is the measure where we excel but need to improve the precision of the model by identifying and adjusting to other types of obfuscation rather than just summary and verbatim obfuscation. Regarding the system runtime, even our goal is not aiming at efficiency, our software performed at an average level.

Table 4.2: Comparison between our result from PAN2015 and PAN2014 approaches on PAN 2014 training corpus

Obfuscation	PAN 2015 approach			
	Plagdet	Recall	Precision	Granul.
None	0.9812	0.9761	0.9933	1.0048
Random	0.8847	0.8699	0.8999	1.0000
Translation	0.8792	0.9128	0.8481	1.0000
Summary	0.6304	0.4862	0.9739	1.0404
Entire	0.9025	0.8937	0.9164	1.0036
Obfuscation	PAN 2014 approach			
	Plagdet	Recall	Precision	Granul.
None	0.8938	0.9782	0.8228	1.0000
Random	0.8886	0.8581	0.9213	1.0000
Translation	0.8839	0.8902	0.8777	1.0000
Summary	0.5772	0.4247	0.9941	1.0434
Entire	0.8773	0.8799	0.8774	1.0021

In Table 4.3 we present a comparison between our approach and 2014 participants showing a remarkable improvement.

Table 4.3: Our approach compared to the PAN 2014 Official results reported in [62]

Team	PlagDet	Recall	Precision	Granularity	Runtime
Sanchez-Perez15	0.9010	0.8957	0.9125	1.0046	–
Sanchez-Perez14	0.8781	0.8790	0.8816	1.0034	00:25:35
Oberreuter	0.8693	0.8577	0.8859	1.0036	00:05:31
Palkovskii	0.8680	0.8263	0.9222	1.0058	01:10:04
Glinos	0.8593	0.7933	0.9625	1.0169	00:23:13
Shrestha	0.8440	0.8378	0.8590	1.0070	69:51:15
R. Torrejón	0.8295	0.7690	0.9042	1.0027	00:00:42
Gross	0.8264	0.7662	0.9327	1.0251	00:03:00
Kong	0.8216	0.8074	0.8400	1.0030	00:05:26
Abnar	0.6722	0.6116	0.7733	1.0224	01:27:00
Alvi	0.6595	0.5506	0.9337	1.0711	00:04:57
Baseline	0.4219	0.3422	0.9293	1.2747	00:30:30
Gillam	0.2830	0.1684	0.8863	1.0000	00:00:55

4.2 Genetic-Based Parameter Tuning

In the majority of works on plagiarism detection, the parameters of the proposed models are determined through brute force [70, 73] or by the author's domain knowledge [32]. Brute force approaches attempt to test all combinations of parameters at once, which is unfeasible if the approach has an extensive set of parameters. The majority of plagiarism detection algorithms apply brute force approaches for smaller subsets of parameters, disregarding plenty of parameter combinations. On the other hand, the parameter setting based on the author's knowledge does not take into account unknown information contained in the data.

There are still few research works that use meta-heuristic based on genetic algorithms for plagiarism detection. Lange and Mancoridis [34] introduced a genetic algorithm for source code plagiarism detection. They defined 18 source code metrics to characterize a developer style and identified the optimal combination of these metrics using a genetic algorithm. The nearest neighbor classifier was used to determine if a piece of code was written by a particular developer. The cited research paper reports that the system is capable of identifying the true author of a source code with 55% of accuracy. However, the main contribution of the paper is the reduction of the search time of the optimal metrics set from weeks to hours by using the genetic algorithm instead of a greedy search.

Bouronara et al. [10] presented an approach for automatic plagiarism detection in the world of mail service based on a machine learning tool and genetic algorithms. Their first approach is based on character n-gram for the representation of the texts and tf-idf as weighting scheme to calculate the importance of a term in the corpus and a combination of two machine learning methods (C4.5 and KNN algorithms). Then, they simulated a meta-heuristic method based on genetic algorithms with some variations of the hyper-parameters. The method based on genetic algorithms improved their initial results by 8.1% in the F-measure score.

Our previews research works [71, 73] introduced a plagiarism detection system that extracts sentences and compares them in a Vector Space Model (VSM) using the cosine similarity alike [32]. It also uses the Dice coefficient as in [33] given that this measure favors an equal vocabulary distribution employed in the passages to be compared. For the extension step, our algorithm clusters together nearby seeds. A plagiarism case is defined by the edges of a cluster and not as a set of seeds, this allows for small gaps in the range of seeds, which can be part of the plagiarism case even if the seeding process did not detect them. When filtering overlapped cases, we proposed a measure of quality to decide which one to keep and which one to discard.

In this work, we introduce a genetic algorithm to optimize our plagiarism detection system [71], tailoring to specific kinds of obfuscation. This optimization allows us to approximate the optimal parameter setting, taking into consideration all the parameters at once. We also seek to reduce time by finding the optimal parameter for our plagiarism detection method. But, unlike Lange and Mancoridis' work [34], our search space is bigger, in the range of 24 trillion (because we are not representing the metrics as binary genes).

We only focus on the seeding and extension components which are the core of our system and where the parameters being optimized are used.

4.2.1 Parameter Tuning

With the objective of optimizing the parameters of our plagiarism detection system in mind, we implemented a genetic algorithm using the basic operations with the settings that we think best fitted our needs. The parameters we tried to optimize are described in Table 4.4. Our implementation of the genetic algorithm starts with a randomly generated population, then applies a fitness function to every individual and finally through the crossover and mutation genetic operators produces a new population. We used the elitist selection when constructing a new population allowing the two best individuals to carry over to the next generation, unaltered; and thus, ensuring that the solution quality of the genetic algorithm does not decrease from one generation to the next. The rest of the population is generated through crossover and mutation. We stop iterating when the maximum number of generations is reached.

4.2.2 Search Space

During the construction of our plagiarism detection system, we extensively experimented with several parameter settings. The results, joined with knowledge of the plagiarism detection field and datasets available, led us to establish certain ranges for each parameter, that we think comprised the best results and might generalize our plagiarism detection model to other datasets.

Table 4.4: Plagiarism detection system parameters being optimized

	Parameter	Description
1	th_cos	Threshold for the cosine similarity during seeding.
2	th_dice	Threshold for the dice coefficient during seeding.
3	th_val	Threshold for cosine similarity during validation.
4	src_size	Min amount of sentences in source fragment.
5	src_gap	Max gap between sentences in source fragment.
6	src_gap_summary	src_gap for the summary detection method.
7	min_src_gap	Min value src_gap can take after several iterations.
8	susp_size	Min amount of sentences in suspicious fragment.
9	susp_gap	Max gap between sentences in suspicious fragment.
10	susp_gap_summary	susp_gap for the summary detection method.
11	min_susp_gap	Min value susp_gap can take after several iterations.

In Table 4.5 we show these domains which define the search space for the genetic algorithm and represents 24,761,352,699,900 possible combinations.

Table 4.5: Gene's domains

Parameter	Domain		
	Min	Max	Step
th_cos	0.20	0.50	0.01
th_dice	0.20	0.50	0.01
th_val	0.20	0.50	0.01
src_size	1	3	1
src_gap	0	30	1
src_gap_summary	0	30	1
min_src_gap	0	9	1
susp_size	1	3	1
susp_gap	0	30	1
susp_gap_summary	0	30	1
min_susp_gap	0	9	1

4.2.3 Crossover

There are many ways the crossover operator may be applied depending on the application and nature of the parameters being optimized [53]. Some of the most well-known operators are:

- Single-point crossover: A single crossover position is chosen at random, and the parts of two parents after the crossover position are exchanged to form two offspring.
- Two-point crossover: Two positions are chosen at random, and the segments between them are exchanged.
- Uniform crossover: The exchange happens at each gene position with a probability p.

Single-point and two-point crossovers cannot represent certain schemes, and genes' ordering is relevant for them. Given that our parameters (genes) lack a natural ordering, we used the uniform crossover in our experiments. The probability p of selecting a gene from a parent is randomly assigned every time the crossover operator is called. We only use two parents which are selected given their fitness function value $w_i = f(x_i)$. Given a population of n individuals $x_1, x_2, \ldots, x_n$ the probability to select an individual x_i for crossover is defined as

$$T = \sum_{j=1}^{n} w_j, \tag{4.1}$$

$$p(x_i) = \frac{w_i}{T}. \tag{4.2}$$

An efficient algorithm to implement the crossover parents selection is given at Algorithm 2.

Algorithm 2: Crossover parents selection

```
1 $T \leftarrow \sum_{j=1}^{n} w_j$
2 $r \leftarrow rand(0, T)$
3 for $i \leftarrow 1, \ldots, n$ do
4     if $r < w_i$ then
5         return $x_i$
6     $r \leftarrow r - w_i$
7 return $x_n$
```

This implementation represents a different equation,

$$p(x_i) = (1 - p_{i-1}) \left(\frac{f(x_i)}{T - f(x_{i-1})} \right), \tag{4.3}$$

which is equivalent to (4.2) and we define as Theorem 1. This theorem can be easily proofed using mathematical induction.

Theorem 1. *The probability of selecting an individual x_i (4.2) and its implementation (4.3) are equivalent.*

$$\frac{f(x_i)}{T} = (1 - p_{i-1}) \left(\frac{f(x_i)}{T - f(x_{i-1})} \right). \tag{4.4}$$

Proof. For $x_0 = 0$, assume $f(x_0) = 0$. Consider the base case $i = 1$; then

$$\frac{f(x_1)}{T} = (1 - 0)\left(\frac{f(x_1)}{T - 0}\right) = \frac{f(x_1)}{T};$$

thus the conclusion holds for $i = 1$. By the inductive hypothesis, assume that the conclusion holds for all values of i up to some k, $k \geq 1$, and consider $i = k + 1$. Then

$$\frac{f(x_{k+1})}{T} = (1 - p_k)\frac{f(x_{k+1})}{T - f(x_k)} \tag{4.5}$$

$$= \left(1 - \frac{f(x_k)}{T}\right)\frac{f(x_{k+1})}{T - f(x_k)} \tag{4.6}$$

$$= \frac{\cancel{T - f(x_k)}}{T}\frac{f(x_{k+1})}{\cancel{T - f(x_k)}} \tag{4.7}$$

$$= \frac{f(x_{k+1})}{T}, \tag{4.8}$$

where (4.6) holds by the inductive hypothesis. $\square$

4.2.4 Mutation

We apply the mutation operator to each gene in every individual in the new population, except for those individuals that moved to the next generation through elitism selection. The mutation depends on a certain rate and when happening it changes the value of a parameter by randomly selecting one of the possible values for that particular parameter.

4.2.5 Fitness Function

The fitness function of the genetic algorithms consists of two parts. At first, we run our plagiarism detection model explained before, and then we compute the *plagdet* metric that returns a value between 0 and 1. The organizers of PAN introduced this metric in the context of their benchmarking workshop [61, 65], which have been in use since 2009 and became the baseline in plagiarism detection evaluation.

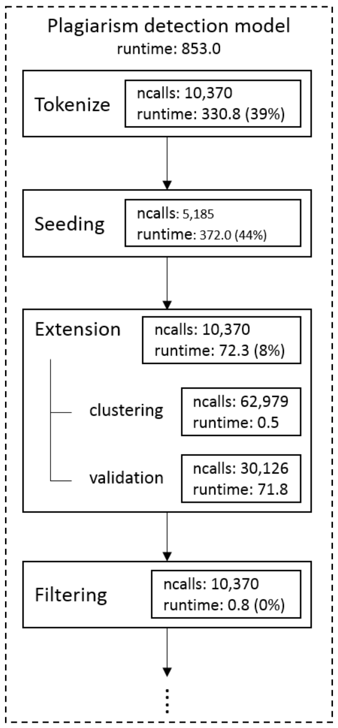


Figure 4.2: System components

4.2.6 Improving the Running Time

PAN 2014 organizers provided an on-line experimentation platform called TIRA [22] where participants were able to submit their systems. In the platform, authors were provided a virtual machine with 1 processor and 4 GB of RAM memory. The running time of our system reported at [62] was of approx. 25 min, something that makes using the system as the fitness function in the genetic algorithm unfeasible. Therefore, we proposed some modifications to the system focusing on the main components of the implementation and its running time as shown in Figure 4.2. Ellipsis stands for the rest of the system we are not worried about its running time. The running time information was obtained using Python's profiler module.

First, we extract the *Tokenize* module from the genetic algorithm which includes all the documents' preprocessing and load all the document BOW representations into memory. Besides *Tokenize*, the most time-consuming module is *Seeding*, which has a complexity of $O(n^2)$ where n is the number

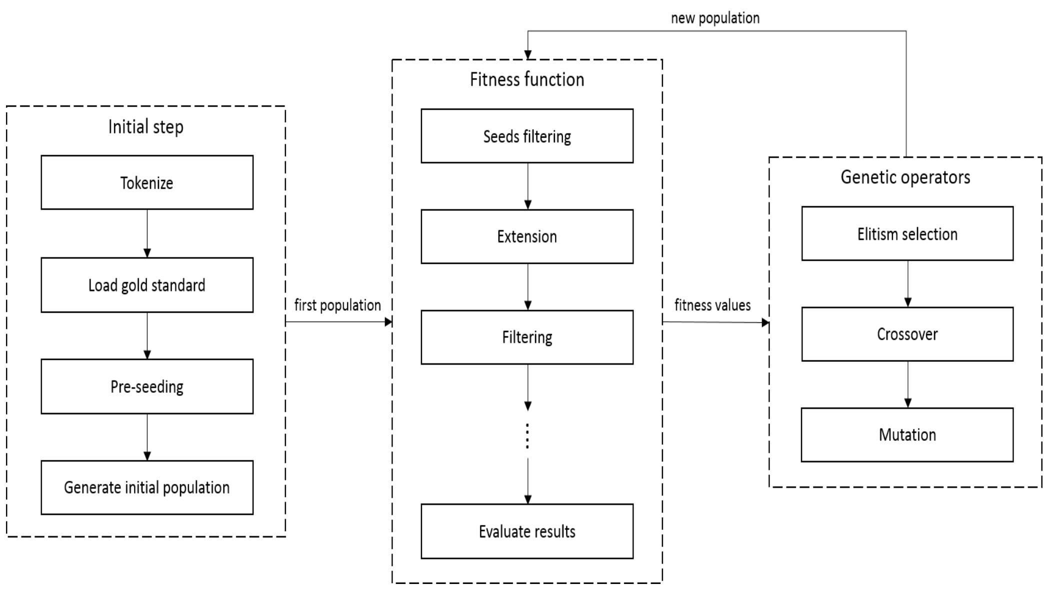


Figure 4.3: Implementation of the genetic algorithm

of sentences in each document. To reduce this running time we compute the seeds for each pair of documents only once using the lowest values of th_cos and th_dice outside the genetic algorithm and load the results into memory. Then, for each call to the fitness function, that calls our plagiarism detection model, we filter the seeds using the new th_cos and th_dice thresholds. We call this process *Seeds filtering*, and the complexity is also $O(n^2)$ for the worst case. However, the worst case is extremely unlikely because given two documents d_1 and d_2, all sentences in d_1 should be similar to the rest of its sentences and all sentences in d_2, and vice-versa.

The diagram of the genetic algorithm is shown in Figure 4.3. During our experiments, we called the fitness function 40,000 times when using the entire corpus and 20,000 times for each of the four sub-corpus.

4.2.7 Experimental Setting

Defined the genetic algorithm operators and the plagiarism detector optimizations, we established some hyper-parameters as shown in Table 4.6. The individual size is the amount of parameters being optimized, and it is fixed. The population size and number of generations were chosen according to the

Table 4.6: Genetic algorithm's hyper-parameters

Number of generations	1000
Population size	20
Individual size	11
Uniform crossover ratio p	$0 \leq p \leq 1$
Mutation rate	0.1

amount of time we had to run tests using two computers with four processors each one. The uniform crossover ratio may have values between 0 and 1, raising the issue of not having the crossover at all for values close to 0 or 1, with a $\frac{2}{18} \approx 0.18$ probability of happening given the individual size used. In future work, we plan to use a different crossover ratio, although this issue did not affect negatively the results given the number of generations and the speed of convergence for this particular application and dataset. The mutation rate was chosen at 0.1, which is a rather large value. This rate was chosen because of the size of each individual, the non-binary representation, and the elitism selection; meaning that at least one gene is changed for every individual. We also have a huge searching space, and we keep the quality of the solution by passing the best individuals to the next generation unaltered.

For our experiments, we used the PAN 2014 training corpus which is divided in five sub-corpus given the type of obfuscation used to generate each plagiarism case. We present a brief definition of each obfuscation type while a detailed description can be found at [63]. It is important to point out that plagiarism cases are corresponding text fragments which are small parts of a given pair of suspicious and source document, meaning we need to find then in a given context and it is not a classification task.

- **No plagiarism:** There are no plagiarism cases in this sub-corpus.
- **None:** Plagiarized fragments are a verbatim copy of the source.
- **Random:** Plagiarized fragments are a randomly obfuscated version of the source.
- **Translation:** Plagiarized fragments are generated by translating a source fragment through several languages and using different machine translators returning to the initial language.

- **Summary:** Plagiarized fragments are human-generated summaries of the sources.

The corpus composition is shown in Table 4.7 where *Pairs* represents the document pairs to be compared, *Pairs w/ PC* the pairs of documents with at least one plagiarism case and *PC* the plagiarism cases in the sub-corpus.

Table 4.7: PAN 2014 training corpus distribution

Sub-corpus	Pairs	Pairs w/ PC	PC
No-plagiarism	1000	0	0
None	1000	1000	1252
Random	1000	1000	1267
Translation	1000	1000	1250
Summary	1185	238	238
Entire corpus	5185	3238	4007

Intuitively, each type of obfuscation has different properties. Hence, proposing an algorithm capable of generalizing for each type of plagiarism with a fixed set of parameters is nearly impossible. In our previous approaches, we tried to address this issue. In PAN 2014 [71] we ran our model twice with different gap parameters, one with *(src_gap, susp_gap)* and the other with larger parameters *(src_gap_summary, susp_gap_summary)*. The aim was to detect between the *summary* sub-corpus and the rest, where the ratio of the size of the plagiarized fragment and the corresponding source was a lot great in this sub-corpus. In [72], we incorporated a longest common substring method that ran over the results of our model and helped to improve the precision of our approach in the *none* sub-corpus dramatically, without hindering the performance of the others. The main idea of this work is to optimize the parameters of the basic model for each one of the sup-corpus, something that could be used later in conjunction with a rule-based [72] or machine learning classifier [43] to improve the performance of the plagiarism detection model.

4.2.8 Results and Discussion

Figure 4.4 shows the best individuals over all generations. We can see that all of the sub-corpus converges quickly to a stable optimal value around the 100th generation. This behavior reflects two things: first, our model

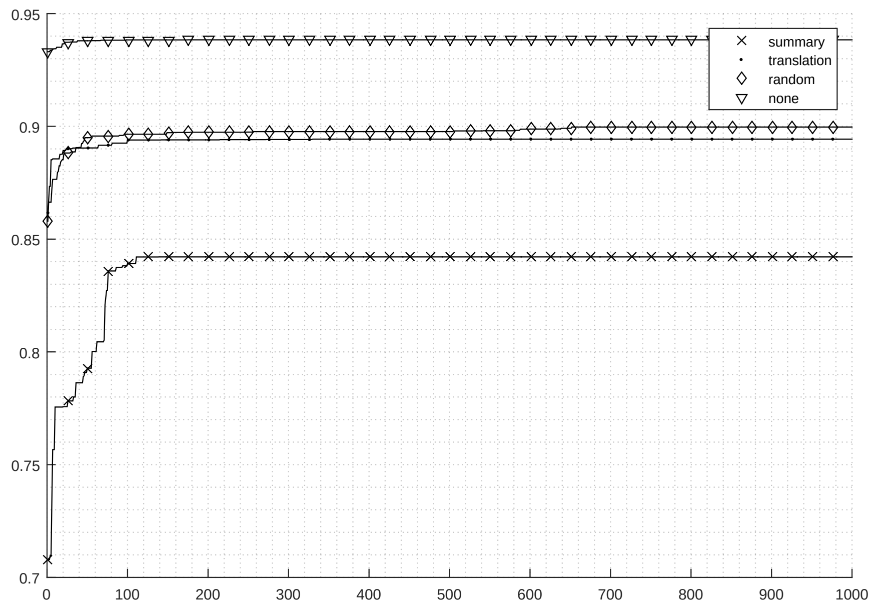


Figure 4.4: Best individual's fitness value by generation

generalizes well for the dataset at hand giving good results for a broad range of parameter values, mostly due to the robustness of the extension algorithm that adjusts some of the parameters dynamically. Second, the fact that the plagiarism cases are generated automatically for the majority of the cases and inserted randomly in a suspicious document increases the probabilities that the surrounding contexts to the plagiarism case are entirely unrelated, like a white box on a black background, allowing simple BOW models to detect the plagiarism cases easily.

The benefit of using a genetic algorithm to optimize the parameters of our model is that allows us to explore several regions of the immense search space taking into account all the parameters at once, something unfeasible to perform using a brute force approach. In Figure 4.5 and Figure 4.6 we plot the distribution of the parameters used as input to our plagiarism detection system as part of the fitness function, i.e. all the parameter values that were tested throughout the 1000 generations. The results show how the parameters converge to certain optimal values depending on the sub-corpus

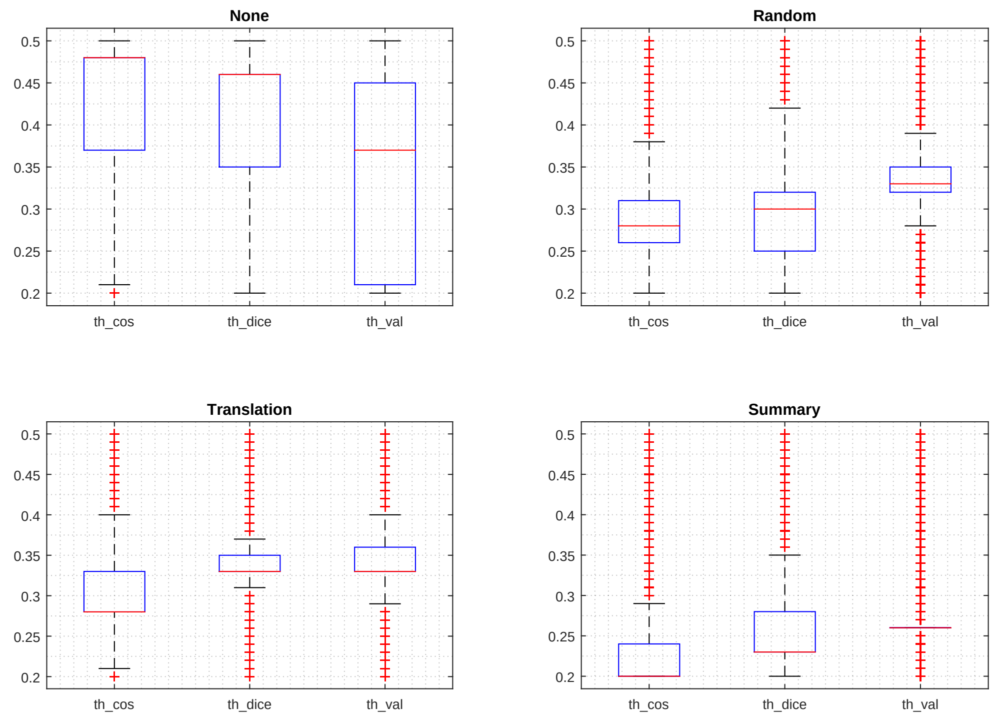


Figure 4.5: Similarity thresholds value's distribution

being used. This convergence happens due to the elitism and the crossover parents selection that keeps the parameter values close to a local optimum.

A deeper analysis of Figures 4.5 and 4.6 shows that for the *None* sub-corpus the threshold th_cos and th_dice, controlling the seeding component, converge to higher values while the th_val does not approach an specific value. This is something to expect given that the plagiarism cases in the *None* sub-corpus are verbatim copies of the sources and hence the similarity of their sentences is close to 1. Likewise, the susp_gap and src_gap tends to have the minimum value because consecutive sentences in plagiarized fragments have exact matches in the source. It is expected to get better results in this sub-corpus; however, the majority of the undetected cases or noise were due to sentence-splitting errors because of missing ending points, that in turn is caused by the random position where a plagiarized fragment was inserted in a suspicious document when generating this sub-corpus.

The *Random* and *Translation* sub-corps results behave similarly given

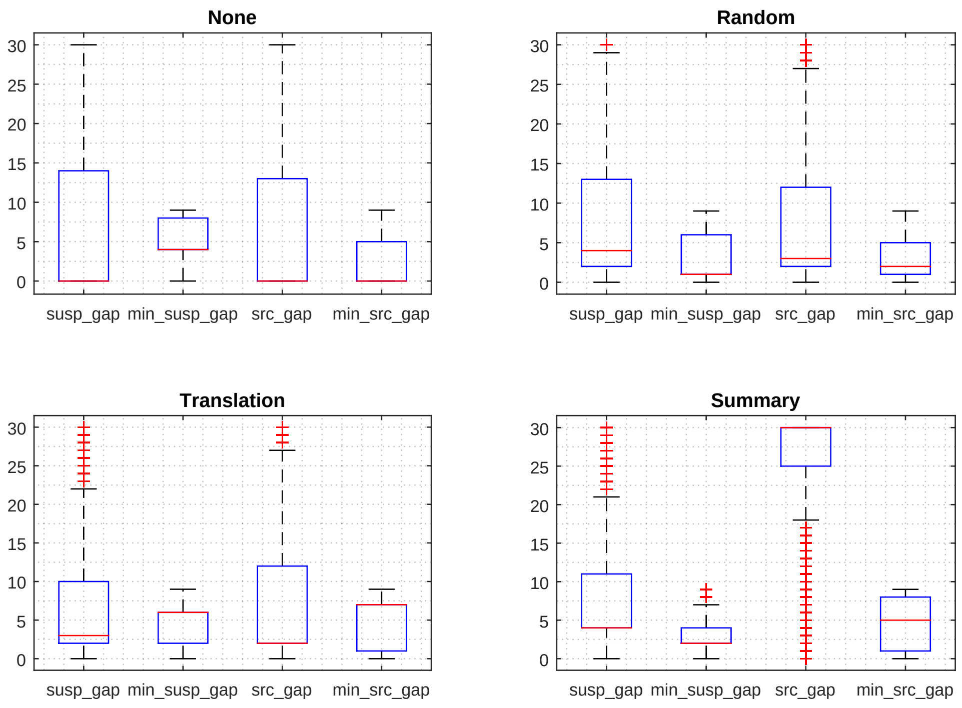


Figure 4.6: Extension parameters value's distribution

that both types of obfuscation simulate paraphrase by changing words with synonyms, sentence reordering or cyclic translation. The similarity measures used in our model does not capture semantic equivalences found in paraphrased sentences, so it relies on lower similarity thresholds for the seeding stage and bigger validation threshold for the extension. An intuitive idea to improve the results for these plagiarism cases is using semantic similarity measures. However, an important note about these two sub-corpus is that in the case of randomly generated cases, the degree of obfuscation varies from a few random changes to a whole lot of them, generating text fragments without any sense to a human. Likewise, plagiarism cases generated through cyclic translation varies from using a pair of extensively translated languages to an abundance of unrelated pairs of languages and naive machine translation approaches, again producing meaningless text fragments.

Finally, in the *Summary* sub-corpus the results reflect the expected behavior of the parameters, where the similarity thresholds are close to the minimum values while the gaps between the source seeds are considerably larger than the suspicious counterpart. The low threshold values suggest that we should use summary detection methods for this kind of obfuscation instead of the traditional cosine similarity and dice coefficient. The difference between the src_gap and susp_gap reflects the nature of a summary, which is smaller that the original text.

Besides running the genetic algorithm for each sub-corpus, we also optimized the parameters of our model with and without the summary heuristic and longest common substring method over the entire corpus (Genetic All & Genetic Simpler respectively). These experiments show some improvements over our previous parameter setting. In Table 4.8 we present the parameters used at PAN and the final parameters configurations resulting from the genetic algorithm for each one of the experiments.

Given the final parameters, we compare the results of each experiment in Table 4.9 by running our plagiarism detection system over the PAN 2014 test corpus. Values with (*) were obtained by gathering the results of running our model over each sub-corpus individually knowing a priori the type of obfuscation and what parameter setting to use. To get these results, we should have had a classifier capable of determining the type of obfuscation in a pair of documents and using the corresponding parameter setting.

When optimizing over the entire corpus (Genetic All & Genetic Simpler), we improved the results, even slightly, of our previous implementations (PAN All & PAN Simpler), which were already the best-performing methods in the

Table 4.8: Final parameters

Parameter	PAN	Genetic					
		Simpler	All	None	Random	Translation	Summary
th_cos	0.30	0.33	0.30	0.48	0.26	0.28	0.20
th_dice	0.33	0.39	0.31	0.46	0.25	0.33	0.23
th_val	0.34	0.22	0.34	0.45	0.33	0.33	0.26
src_size	1	1	1	1	1	1	1
src_gap	4	19	3	0	2	2	30
src_gap_summary	24	-	29	-	-	-	-
min_src_gap	0	9	3	0	1	2	8
susp_size	1	1	1	1	1	1	3
susp_gap	4	4	3	0	2	2	4
susp_gap_summary	24	-	28	-	-	-	-
min_susp_gap	0	0	3	0	1	2	2

Table 4.9: Plagiarism detection results using the final parameters over PAN 2014 test corpus

	None	Random	Translation	Summary	Entire
PAN All	0.9854	0.8846	0.8751	0.6329	0.9010
Genetic All	**0.9859**	0.8841	0.8764	0.6493	0.9021
Genetic	0.9421	**0.8921**	**0.8940**	**0.8192**	0.9041*
Genetic + LCS	0.9732	0.8757	0.8936	0.8173	**0.9085***
PAN Simpler	**0.9010**	**0.8912**	**0.8868**	0.3108	0.8687
Genetic Simpler	0.8960	0.8751	0.8763	**0.6261**	**0.8733**

PAN 2014 Text Alignment corpus [62].

As expected, specific obfuscation type optimization gave better results than those obtained with the fixed set of parameters, except for the *None* sub-corpus which performed better in the approaches that use the Longest Common Substring (LCS) method. Hence, we also added the results of using the parameters of the genetic algorithm with the LCS method, which outperformed the rest of the experiments.

Chapter 5

Paraphrase Recognition

5.1 Basic Approach

Our first approach to paraphrase recognition is applying different techniques used in plagiarism detection to asses its effectiveness in the task. We evaluate this first approach in the P4P corpus which is tagged following the typology presented in section 3.2.1. For the experiments, we proposed several sets of settings depending on the pre-processing steps, the features extracted, the weighting scheme, and the similarity measure.

We decided to use term frequency as weighting scheme and cosine measure to compute the similarity between pair of sentences in the P4P+N corpus. The P4P+N corpus is an addition of 2539 non-plagiarism cases to the regular P4P corpus. It is important to note that given that the dataset is unbalanced toward the negative cases, with just 847 positives paraphrase, the F1 measure is biased toward precision.

We experimented selecting words as features (Table 5.1) with combinations of stemming and two ways of removing stop words:

1. 50 most frequent stop words (rem. 50 sw) in BNC corpus,

2. All the words reported in Brown corpus (rem. all sw).

We also experimented with the following features: unsorted and sorted n-grams (5.2 and 5.3), stop words n-grams 5.4, and skip n-grams 5.5. However, to avoid cluttering this section with too many results we just show the results of the without the combinations of preprocessing steps.

We can appreciate that commonly used approaches to plagiarism detection do not work in paraphrase recognition. The results show that the best approach is using the basic bag of words model and a threshold of 0.25 in the cosine similarity when classifying a case as plagiarism/paraphrase. For that specific instance, precision is 0.876 and recall 0.855. For the other results, the trade-off between precision and recall change dramatically with one of the measurements rapidly going towards 0 or 1. For this basic approach we do not perform any experiment on the MSRPC, instead we move to more elaborate models based on Knowledge bases.

Table 5.1: F1-score for selected features: words

cos threshold	1	2	3	4	5	6
stemming	✓	✓	✓			
rem. 50 sw		✓			✓	
rem. all sw			✓			✓
0	0.393	0.488	0.508	0.393	0.517	0.543
5	0.396	0.600	0.606	0.396	0.648	0.657
10	0.403	0.715	0.706	0.403	0.762	0.755
15	0.421	0.795	0.790	0.421	0.821	0.825
20	0.449	0.850	0.835	0.449	0.858	0.845
25	0.487	**0.865**	0.858	0.485	0.861	0.847
30	0.546	0.855	0.847	0.544	0.836	0.833
35	0.614	0.829	0.824	0.609	0.801	0.794
40	0.693	0.786	0.788	0.686	0.749	0.746
45	0.759	0.735	0.734	0.748	0.697	0.691
50	0.800	0.682	0.678	0.783	0.624	0.628
55	0.788	0.608	0.609	0.771	0.546	0.548
60	0.738	0.521	0.523	0.708	0.468	0.457
65	0.658	0.418	0.440	0.633	0.369	0.380
			....			

Table 5.2: F1-score for unsorted n-grams with $n = 1...7$.

F1	1-gram	2-gram	3-gram	4-gram	5-gram	6-gram	7-gram
0	0.488	0.798	0.685	0.524	0.393	0.308	0.238
5	0.600	0.783	0.587	0.431	0.314	0.235	0.182
10	0.715	0.740	0.493	0.346	0.255	0.192	0.161
15	0.795	0.677	0.427	0.293	0.206	0.159	0.138
20	0.850	0.611	0.353	0.225	0.172	0.136	0.115
25	**0.865**	0.531	0.288	0.198	0.136	0.117	0.097
30	0.855	0.462	0.240	0.159	0.117	0.1	0.075
35	0.829	0.394	0.198	0.130	0.102	0.070	0.059
40	0.786	0.307	0.161	0.106	0.084	0.059	0.050
45	0.735	0.259	0.128	0.093	0.063	0.052	0.045
50	0.682	0.202	0.104	0.066	0.052	0.040	0.033
			....				

Table 5.3: Sorted n-grams with $n = 1...7$

F1	1-gram	2-gram	3-gram	4-gram	5-gram	6-gram	7-gram
0	0.488	0.799	0.710	0.560	0.435	0.351	0.272
5	0.600	0.801	0.616	0.462	0.333	0.263	0.212
10	0.715	0.755	0.523	0.381	0.272	0.208	0.172
15	0.795	0.697	0.447	0.307	0.219	0.170	0.145
20	0.850	0.627	0.380	0.238	0.176	0.145	0.126
25	**0.865**	0.549	0.304	0.212	0.153	0.121	0.106
30	0.855	0.483	0.252	0.172	0.128	0.104	0.082
35	0.829	0.411	0.210	0.136	0.104	0.077	0.061
40	0.786	0.316	0.168	0.113	0.088	0.064	0.052
45	0.735	0.265	0.136	0.093	0.068	0.052	0.045
50	0.682	0.212	0.110	0.066	0.052	0.040	0.033
			....				

Table 5.4: Stop word n-grams with $n = 6...11$

F1	sw6-gram	sw7-gram	sw8-gram	sw9-gram	sw10-gram	sw11-gram
0	**0.367**	0.318	0.251	0.203	0.181	0.153
5	0.335	0.271	0.226	0.191	0.171	0.146
10	0.292	0.241	0.205	0.181	0.159	0.138
15	0.260	0.222	0.191	0.167	0.151	0.125
20	0.224	0.205	0.177	0.161	0.138	0.119
25	0.201	0.183	0.167	0.142	0.127	0.114
30	0.187	0.175	0.148	0.134	0.127	0.110
35	0.169	0.155	0.140	0.125	0.117	0.104
40	0.159	0.148	0.134	0.117	0.110	0.099
45	0.148	0.140	0.125	0.115	0.106	0.097
50	0.134	0.127	0.117	0.108	0.104	0.086
			....			

Table 5.5: Skip n-grams with $n = 3...7$

F1	3-gram	4-gram	5-gram	6-gram	7-gram
0	**0.725**	0.558	0.422	0.315	0.233
5	0.609	0.436	0.316	0.229	0.180
10	0.504	0.348	0.242	0.186	0.159
15	0.440	0.270	0.198	0.157	0.141
20	0.363	0.229	0.163	0.143	0.113
25	0.283	0.186	0.147	0.119	0.088
30	0.233	0.157	0.117	0.088	0.068
35	0.188	0.128	0.091	0.068	0.064
40	0.157	0.102	0.073	0.061	0.050
45	0.119	0.077	0.061	0.052	0.045
50	0.095	0.066	0.052	0.045	0.036
			....		

5.2 Knowledge-Based Model

There are some models that extend the capabilities of the vector space model and cosine metric from just computing lexical similarity. Specifically, there are some approaches to paraphrase identification that use Knowledge bases to compute semantic similarity **add citations Rada, Eisenstein**. We propose some experiments using combinations of these models and adding a new metric called Softcosine proposed by Grigori et al [77].

We conducted a series of experiments for several combination of parameters grouped by some of the following areas:

- Preprocessing parameters
  - Removing stopwords
  - Removing punctuation
- Feature extraction parameters
  - Features extracted: [tokens, lemmas, stems, lemmas+pos]
  - Weighting scheme: [tf, binary (bin)]
- WordNet parameters
  - Features used to retrieve the synsets: [tokens, lemmas, lemmas+pos]
  - Synsets selection. There are two possible strategies when computing the similarity between two features:
    - (1) We compute the similarity only between the first synset of each term
    - (n) We take the maximum similarity resulting of comparing all the synsets in both terms
  - WordNet similarity metric
    - (path): Path
    - (lch): Leacock & Chodorow [35]
    - (wup): Wu & Palmer [86]
    - (res): Resnik [68]
    - (jcn): Jiang & Conrath [29]
    - (lin): Lin [38]

  - Information content. Some WordNet similarity metrics uses Information Content extracted from a given corpus:
    * (bnc07): 2007 British National Corpus
    * (bnc00): NLTK 2000 British National Corpus
    * (brown): NTLK Brown corpus
    * (semcor): NTLK Semcor corpus
  - WordNet normalization: Some WordNet similarity metrics gives values outside the range [0,1]
  - Terms similarity threshold: We set to zero all term-term similarity values below this threshold

- Sentence similarity parameters

- Text similarity metric: We use three type of computing the similarity between two text fragments: [mihalcea [47], stevenson [17], softcosine [77]]

- Text similarity threshold: Everything above this threshold is consider a case of paraphrase

In Table5.2 and Table5.2 we present the best results for each text and WordNet similarity metrics, sorted by accuracy and f1-score respectively. We omit the preprocessing steps from the table because we get the best results without removing stopwords and eliminating punctuation symbols.

Given the results on the training dataset, we use these configurations to predict the performance in the test dataset as shown in Table5.2.

Table 5.6: Best accuracies for all possible combinations in training dataset

Features	Weighting scheme	Synset feature	Synsets selection	WordNet metric	Information content	WorNet normalization	Terms sim. min th	Text sim. metric	Text sim. th	Accuracy	F1-score
lemma	tf	lemma	n	lin	bnc07		0.50	mihalcea	0.70	0.741	0.819
stem	tf	token	n	path			0.50	mihalcea	0.60	0.739	0.821
lemma	tf	lemma	n	res	brown	✓	0.00	mihalcea	0.65	0.737	0.817
stem	bin	token	n	lin	bnc00		0.75	softcosine	0.65	0.736	0.813
stem	tf	token	n	jcn	brown	✓	0.50	mihalcea	0.55	0.735	0.824
token	tf	lemma	n	lch		✓	0.00	mihalcea	0.70	0.735	0.821
stem	bin	token	n	jcn	semcor	✓	0.00	stevenson	0.55	0.734	0.826
stem	bin	token	n	path			0.50	softcosine	0.60	0.734	0.817
token	bin	token	1	lin	semcor		0.75	stevenson	0.55	0.733	0.824
stem	tf	token	n	wup			0.75	mihalcea	0.65	0.733	0.821
stem	bin	token	1	res	bnc07	✓	0.25	softcosine	0.55	0.733	0.823
stem	bin	token	1	path			0.50	stevenson	0.55	0.733	0.821
token	bin	token	1	jcn	brown	✓	0.00	softcosine	0.55	0.732	0.819
lemma	bin	lemmapos	n	lch		✓	0.75	softcosine	0.55	0.732	0.820
stem	bin	token	1	lch		✓	0.75	stevenson	0.55	0.732	0.819
stem	bin	lemmapos	1	wup			0.75	softcosine	0.55	0.730	0.819
lemma	bin	lemma	n	res	brown	✓	0.75	stevenson	0.55	0.730	0.816
stem	bin	lemmapos	1	wup			0.75	stevenson	0.55	0.727	0.818

Table 5.7: Best f1-scores for all possible combinations in training dataset

Features	Weighting scheme	Synset feature	Synsets selection	WordNet metric	Information content	WorNet normalization	Terms sim. min th	Text sim. metric	Text sim. th	Accuracy	F1-score
stem	bin	lemmapos	1	res	semcor	✓	0.00	mihalcea	0.55	0.733	0.828
stem	tf	lemma	1	lin	bnc00		0.50	mihalcea	0.55	0.733	0.828
stem	bin	token	1	lin	semcor		0.75	stevenson	0.55	0.733	0.825
stem	bin	lemma	1	lin	bnc00		0.50	softcosine	0.55	0.729	0.825
stem	tf	lemma	1	path			0.00	mihalcea	0.55	0.728	0.826
stem	bin	lemma	n	res	brown	✓	0.75	stevenson	0.50	0.727	0.825
stem	bin	lemma	n	jcn	bnc07	✓	0.00	softcosine	0.50	0.726	0.826
stem	bin	lemmapos	1	res	bnc07	✓	0.25	softcosine	0.50	0.726	0.826
stem	tf	lemma	n	jcn	bnc07	✓	0.75	mihalcea	0.50	0.725	0.826
token	tf	lemma	n	lch		✓	0.75	mihalcea	0.50	0.725	0.825
stem	bin	token	n	jcn	semcor	✓	0.25	stevenson	0.50	0.724	0.826
stem	bin	token	n	path			0.75	softcosine	0.50	0.723	0.825
stem	bin	lemmapos	1	wup			0.75	mihalcea	0.50	0.722	0.824
stem	bin	token	n	lch		✓	0.75	softcosine	0.50	0.722	0.825
stem	bin	token	n	path			0.75	stevenson	0.50	0.722	0.825
stem	bin	lemmapos	1	wup			0.75	softcosine	0.50	0.720	0.822
stem	bin	lemma	n	lch		✓	0.75	stevenson	0.50	0.720	0.824
stem	bin	lemmapos	1	wup			0.75	stevenson	0.50	0.716	0.821

Table 5.8: Accuracy and F1-score using best combinations on training dataset and evaluated on test dataset

Text sim.	WordNet metric	Accuracy	F1-score
stevenson	path	0.743	0.826
mihalcea	jcn	0.741	0.826
stevenson	lch	0.739	0.823
softcosine	path	0.738	0.817
mihalcea	path	0.737	0.818
stevenson	jcn	0.736	0.824
softcosine	jcn	0.736	0.819
softcosine	wup	0.736	0.821
softcosine	res	0.735	0.824
softcosine	lch	0.735	0.821
stevenson	res	0.734	0.818
mihalcea	res	0.733	0.813
stevenson	wup	0.732	0.820
mihalcea	wup	0.730	0.815
stevenson	lin	0.729	0.820
softcosine	lin	0.726	0.803
mihalcea	lch	0.725	0.812
mihalcea	lin	0.724	0.803

Table 5.9: Comparison of our experiments against the reported results

Method	WordNet Metric	Accuracy	F1-score
Mihalcea [47]	jcn	69.3	79.0
Mihalcea [47]	lch	69.5	79.0
Mihalcea [47]	Combined	70.3	81.3
Our mihalcea	jcn	**74.1**	**82.6**
Fernando & Stevenson [17]	jcn	74.1	82.4
Our stevenson	jcn	73.6	82.4
Our stevenson	path	**74.3**	**82.6**
Our softcosine	path	**73.8**	81.7
Our softcosine	jcn	73.6	81.9
Our softcosine	wup	73.6	82.1
Our softcosine	res	73.5	**82.4**
Our softcosine	lch	73.5	82.1

5.3 Distributional Models

In this section we analyzed some approaches in depth, trying to replicate their results. Specifically, we focus on three paraphrase identification models: He et al. [26], Wang et al. [84], and Shen et al. [75]. We also experimented with Kim's compositional model [31]. The advantages of the paraphrase identification task is that most of the approaches report their results on the Microsoft Research Paraphrase Corpus (MSRPC), which have become the benchmark dataset for this task.

We will describe in more details the selected approaches providing information about the implementations and challenges presented.

5.3.1 Preprocessing and Embeddings Transformation

Given that none of the selected approaches mentioned how they preprocess the corpus, we experimented with two methods: 1) using the implementation of Kim [31][1] which have been used in several deep learning approaches for a wide range of applications, and 2) another proposed by us trying to minimize the amount of errors of the tokenizer.

Preprocessing methods play an important role when testing the models on the MSRPC. This dataset is consider small and it is prone to tokenization errors due to the large number of unnormalized textual citations. The result of the preprocessing step greatly influence the transformation of word embeddings and the number of unknown tokens not found in the pre-trained embeddings and that have to be randomly initialized.

The most popular pre-trained embeddings available, like word2vec or GloVe, were trained without applying much preprocessing to the data. To reduce the number of unknown words we search for the original token, if not present we look for the token with the first letter capitalized (only when lower casing the text), and finally the lower/upper cased token accordingly. We call this method Advance lookup (Adv. lookup), otherwise Simple lookup (Smp. lookup).

In table Table 5.10 and 5.11, we show some statistics about the number of unknown words when computing the embedding matrix of the MSRPC dataset. We experiment with four pre-trained embeddings, lower casing, and the embeddings retrieving method.

[1]`https://github.com/yoonkim/CNN_sentence`

Table 5.10: Number of unknown words using Kim's preprocessing script

	lower		nolower	
	Adv. lookup	Smp. lookup	Adv. lookup	Smp. lookup
Vocab Size	15802	15802	17523	17523
Word2vec	1215 (8%)	3364 (21%)	1253 (7%)	1260 (7%)
GloVe	504 (3%)	1104 (7%)	488 (3%)	494 (3%)
Paragram25	1844 (12%)	1844 (12%)	1868 (11%)	7248 (41%)
Paragram300	467 (3%)	467 (3%)	476 (3%)	6821 (39%)

Table 5.11: Number of unknown words using our preprocessing script

	lower		nolower	
	Adv. lookup	Smp. lookup	Adv. lookup	Smp. lookup
Vocab Size	15671	15671	17387	17387
Word2vec	1086 (7%)	3233 (21%)	1116 (6%)	1122 (6%)
GloVe	374 (2%)	965 (6%)	349 (2%)	357 (2%)
Paragram25	1701 (11%)	1701 (11%)	1718 (10%)	7124 (41%)
Paragram300	353 (2%)	353 (2%)	355 (2%)	6700 (39%)

5.3.2 Multi-Perspective Sentence Similarity Modeling with CNN

He et al. [26] provide an implementation using Torch and Lua[2]. However, there are some differences with regard to their paper given that this implementation represents their follow up work presented in [27]. They only use GloVe embeddings and provide scripts to run their algorithm on two of the three tasks they mention in their paper, two SemEval semantic relatedness tasks [1, 42], but not the paraphrase identification task. It is important to note that they only use the ∞ kernel size for holistic convolutions. Another difference between the original paper and the available implementation, which is not stated in the follow up paper, is that they only use a two-dimensional convolution filter as the per-dimension convolution instead of *embeddingsdim* independent filters.

We implemented this approach using Keras[3] and ran the experiments using GPU's which considerably sped up the process. In their paper the authors

[2]`https://github.com/hohoCode/textSimilarityConvNet`

[3]`https://keras.io/`

do not specify a preprocessing step and in their implementation they expect a preprocessed and tokenized dataset. Hence, we selected the kim preprocessing method without lowering case, and with our "Advance lookup" method. As shown previously in Table5.10 and 5.11, this combination consistently returns the fewer unknown tokens for the selected pre-trained embeddings.

Given that we do not have access to the corpus they used for training the POS embeddings, we compare our implementation results against those reported in their ablation study without these embeddings.

The main difference between our approach and their implementation is the use of padding. We pad all the sequences to the same length in order to use Keras tensors. This action may have some consequences when applying the pooling operators.

Results and Analysis

One of the main piece of information missing in their paper is a comparison between the training accuracy and validation accuracy that allow us to study the bias-variance tradeoff. We assume training such a complex model on the MSRPC alone with result in overfitting. In Figure5.1 we can appreciate that the training accuracy steadily increases while the validation accuracy fluctuates almost randomly. We also observe this behavior in Figure5.2 representing the loss being optimized, in this particular experiment: cross-entropy with stocastic gradient descent (SGD) and learning rate of 0.001.

Using a different optimizer like Adam, allows the model to converge faster although is does not solve the overfitting problem as shown in Figure5.3 and Figure5.4. More experiments are needed to fine tuned the hyper-parameters. However, we think more efforts should be devoted to train these models on large datasets first.

Additionally, we slightly modified their implementation to run on the MSRP corpus and be able to compare the results of both implementations. In Table5.3.2 we show some details of the implementations and the settings reported in their paper.

Using He's implementation we obtain the predictions for each epoch. Using the predictions we can compute the metrics of the dev dataset to compare to our implementation. In Figure5.5 we observe both approaches behave similarly which shows that our implementation works similar to the provided by the authors. However, running their code in 4 CPU threads (they did not use GPU) took $\approx$ 20 hours, while our approach ran in 6 minutes on one Nvidia

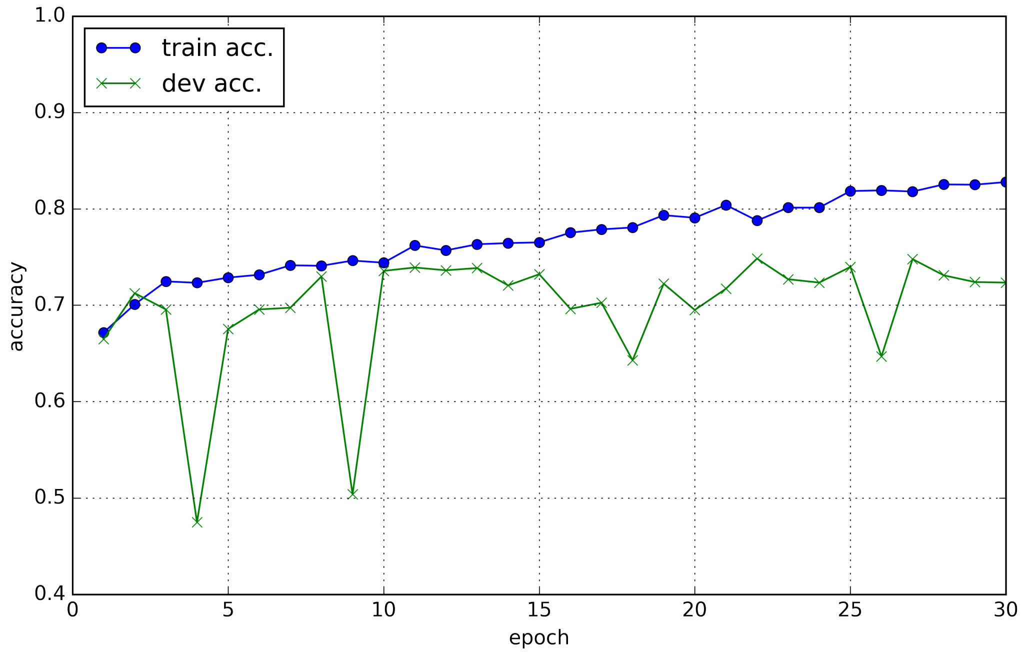


Figure 5.1: He's model. Train vs. Dev accuracy using SGD optimizer (lr=0.01)

Table 5.12: Differences between implementations and described model

	Our impl.	He's impl.	He's model
Embeddings	GloVe	GloVe	GloVe, W2V, paragram
Embeddings dim	300	300	300+200+25=525
Trainable emb.	Yes	No	Yes
Loss	Cross-entropy	KL Divergence	Hinge
Optimizer	Adam	SGD	SGD
Learning rate	0.001	0.01	0.01
Regularization	10^{-4}	10^{-4}	10^{-4}
Hidden units	150	150	250
Parameters	9,503,912	9,503,912	38,841,827
Padding	Yes	No	No

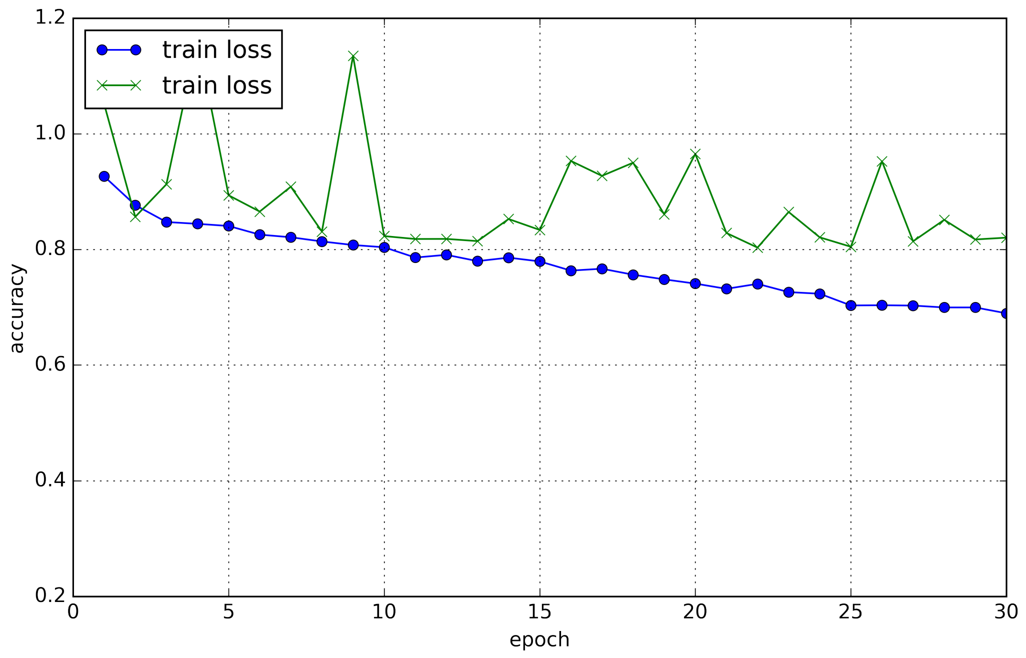


Figure 5.2: He's model. Train vs. Dev cross-entropy loss using SGD optimizer (lr=0.01)

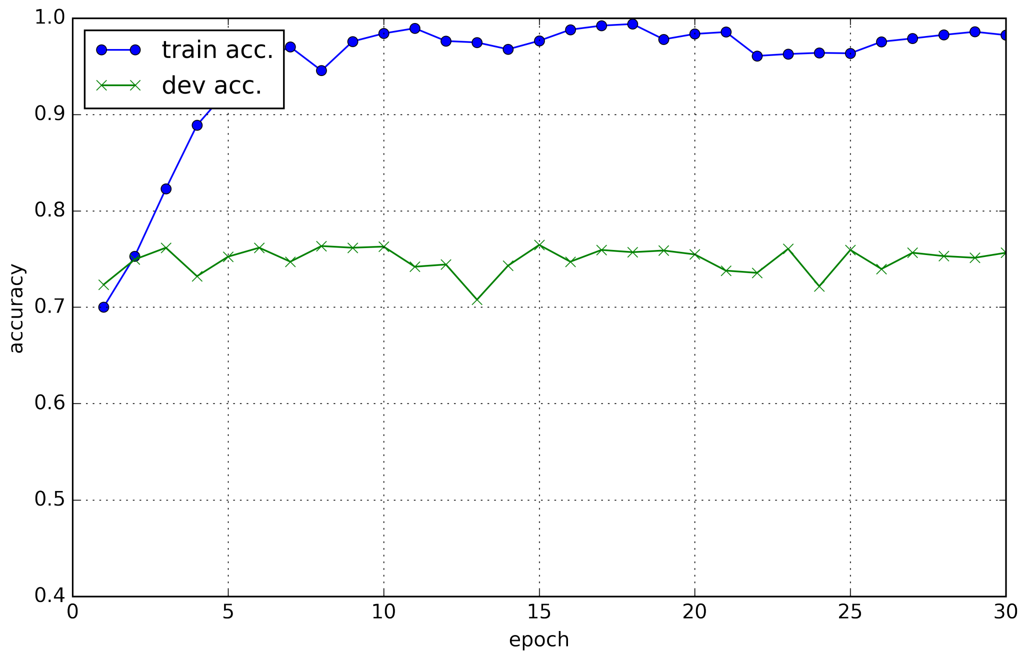


Figure 5.3: He's model. Train vs. Dev accuracy using Adam optimizer (lr=0.001)

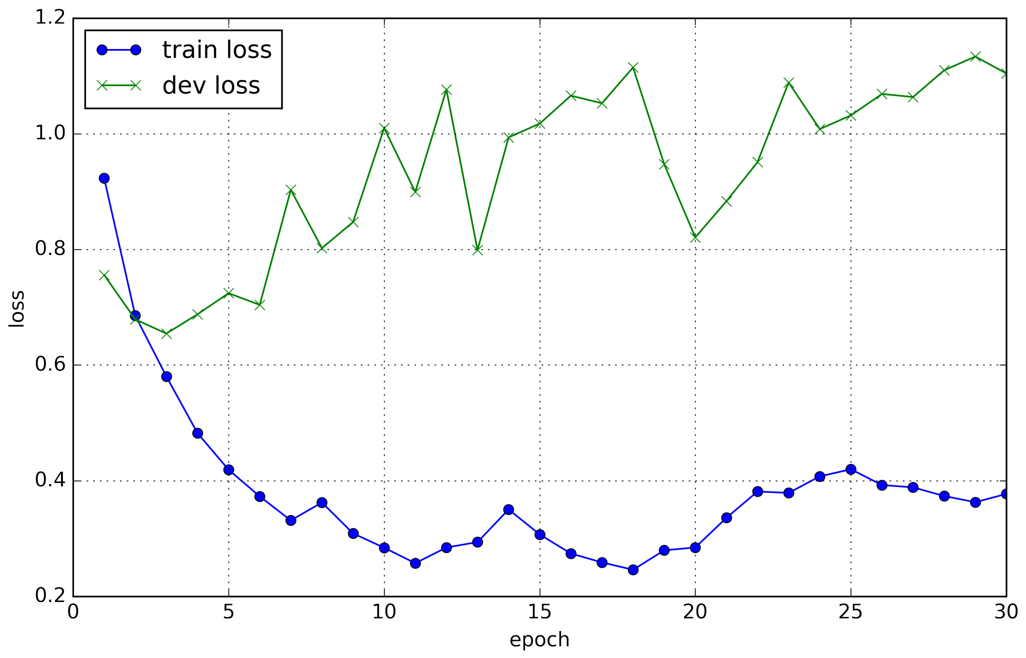


Figure 5.4: He's model. Train vs. Dev cross-entropy loss using Adam optimizer (lr=0.001)

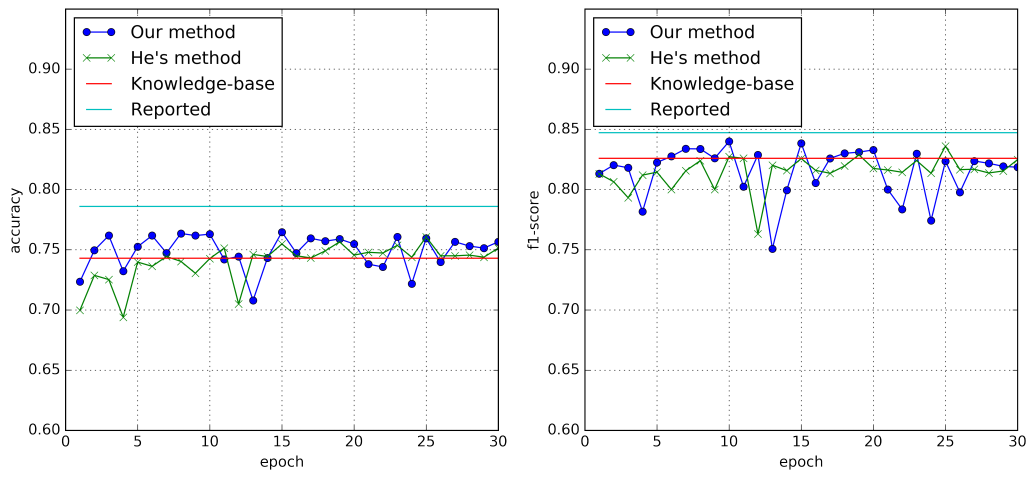


Figure 5.5: Comparison of accuracies and f1-score of ours and He's implementation

GeForce GTX 1080.

We attribute the poor performance to the small size of the MSRPC, which has only 5800 samples. To put into perspective the complexity of the model to the size of the corpus, in the next section we compute the number of parameters being optimized in the model. Given the following settings (Table5.3.2): Then, the number of parameters of the compositional model is

Table 5.13: Experimental settings

Embeddings dim (d)	300
Kernel sizes (W)	{1,2,3}
Holistic filters (f_a)	300
Per-dim. filters (f_b)	20
Holistic pool op. (P_a)	{max,min,mean}
Per-dim. pool op. (P_b)	{max,min}

given by the following equations:

$$p_{holistic} = |P_a| f_a \sum_{w \in W} (wd + 1) \tag{5.1}$$

$$p_{perdim} = |P_b| f_b \sum_{w \in W} (wd + 1) \tag{5.2}$$

$$p_{perdim'} = |P_b| f_b \sum_{w \in W} (w + 1) \tag{5.3}$$

$$total = p_{holistic} + g_{perdim} \tag{5.4}$$

$$total' = p_{holistic} + g_{perdim'}. \tag{5.5}$$

The values of $p_{perdim'}$ and $total'$ stand for the new variation of He's model where they only use one filter for the per dimension convolution instead of $embedding_{dim}$ independent filters. For our particular example,

$$\begin{aligned} p_{holistic} &= 1,622,700 \\ p_{perdim} &= 108,000 \\ p_{perdim'} &= 360 \\ total &= 1,730,700 \\ total' &= 1,623,060 \end{aligned}$$

The output shape of the compositional model are two tensors of shape $(|P_a|, |W|+1, f_a)$ and $(|P_b|, |W|, d, f_b)$, corresponding to the holistic and per-dimension filters respectively.

In the similarity layer, He's algorithms uses two set of vectorial metrics: cosine similarity (scalar) and euclidean distance (scalar); and the previous ones plus the absolute distance (d-dim vector). The output shape of the similarity layer before the last section of fully connected layers is given by (i.e. concatenating all the region comparisons):

$$s = 2 \cdot |P_a| \cdot f_a \tag{5.6}$$

$$+ (2 + d) \cdot |P_a| \cdot (|W| + 1)^2 \tag{5.7}$$

$$+ (2 + d) \cdot |P_b| \cdot |W| \cdot f_b, \tag{5.8}$$

where 5.6 represents the algorithm 1 comparing holistic filter outputs, 5.7 the algorithm 2 comparing holistic filters, and 5.8 the algorithm 2 comparing per-dimension filters.

Table 5.14: Gensim word2vec model's parameters

Training model	Skip-gram
Embeddings size	200
Min term frequency	5
Window size	5

Following our example, $s = 52,536$. Knowing this value, we can compute the number of parameters optimized in the fully connected layer. Assuming a two-layer NN with $h = 150$ hidden units and two output classes, then the number of parameters of this stage is $sh + 3h + 2 = 7,880,852$. Finally, the number of parameters of the last variation of He's model is: $1,623,060 + 7,880,852 = 9,503,912$. This value is extremely large compared to the $5,800$ samples in the MSRP corpus.

5.3.3 Sentence Similarity Learning by Lexical Decomposition and Composition

Wang et al. [84], similar to other approaches, uses pre-trained embeddings. Specifically, they use the 300-dim word2vec embeddings provided by Google. The next step is decomposing each word vector in a sentence pair in a similar and dissimilar component. This process is based on the cosine similarity between the two sequences of embeddings (representing the sentence pair). The decomposition occurs as explained in the compositional model section using the provided formulas. This part of their model do not train any parameters, is only until the compositional part where they use Kim's model [31] adapted to two channel inputs.

The authors do not provide any implementation of their model, however, there is an implementation available on GitHub by the user mcrisc using Tensorflow[4]. The implementation is modulated in parts that need to be ran independently and it is evaluated on trec-qa task only.

Similarly to our implementation of He's model, we use Keras and Numpy, reusing parts of our method, specifically the preprocessing and data preparation modules. This approach to the problem allow us to compare several models under the same circumstances.

The decomposition module is inspired by the available implementation

[4]urlhttps://github.com/mcrisc/lexdecomp

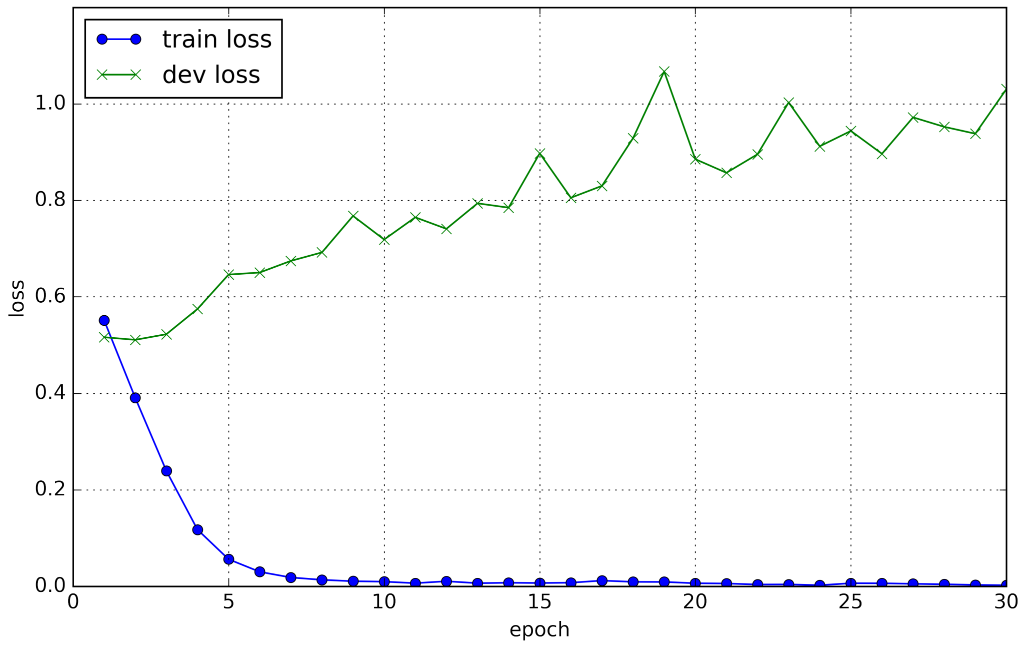


Figure 5.6: Train vs. Dev cross-entropy loss

but improving on the running time by taking advantage of Numpy broadcasting property although irrelevant compared to the training time of the compositional and similarity layers. The composition module is our adaptation of Kim's model to multichannel inputs. There are several implementations available for Kim's approach that served as guidance for our code.

Results and Analysis

In Figure5.6 the dev loss increases contrarily to the train loss that rapidly decreases almost to zero. Similarly, in Figure5.7 the huge gap and the different behavior of train and dev accuracy presents a clear case of overfitting. The stability of the dev accuracy is a product of the first stage of Wang's model where the semantic similarity is computed without learning any parameter. Also, it shows that none of the parameters or features learned in the composition model during training is relevant to the dev dataset, and in conclusion to the paraphrase identification task.

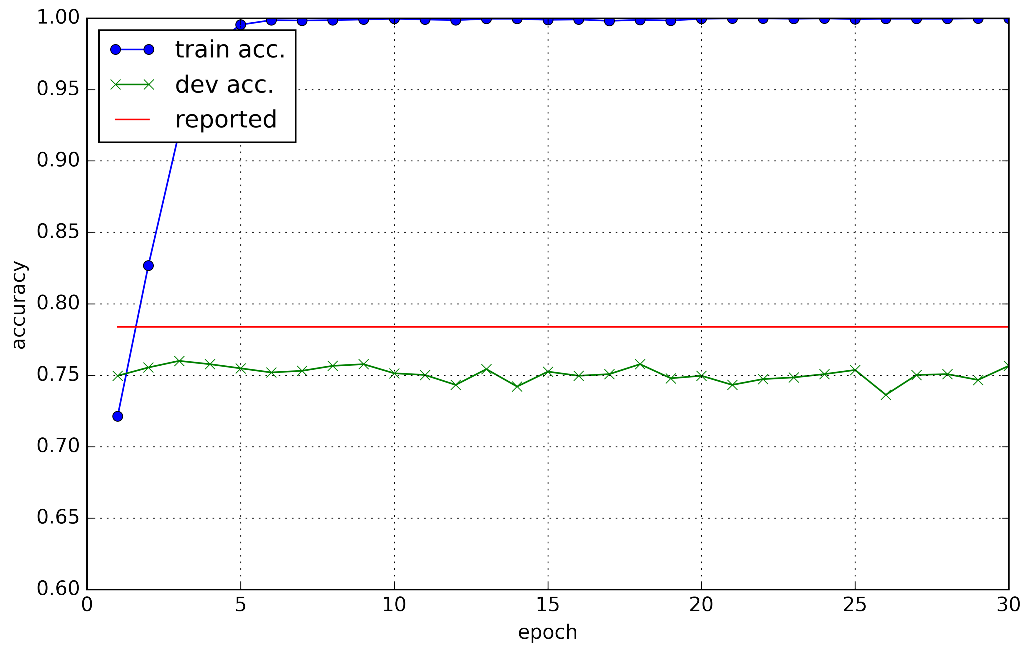


Figure 5.7: Train vs. Dev accuracy

5.3.4 Bidirectional LSTM with Gated Relevance Network

Shen et al. [75] provide a detailed description of their model including several hyperparameters. However, the authors do not provide an open source implementation.

The model starts with pre-trained word2vec embeddings, the sequences of vectors are composed to sentence representations using a bidirectional LSTM with outputs in each time step. Their main contribution is in the similarity layer where they propose a gated relevance network consisting of a bilinear tensor product, a traditional two-layer neural network joined by a gate mechanism.

As there are not available implementations of this model, we implemented the gated relevance network as a Keras custom layer and generated units test validating the forward pass output against a step-by-step numpy implementation.

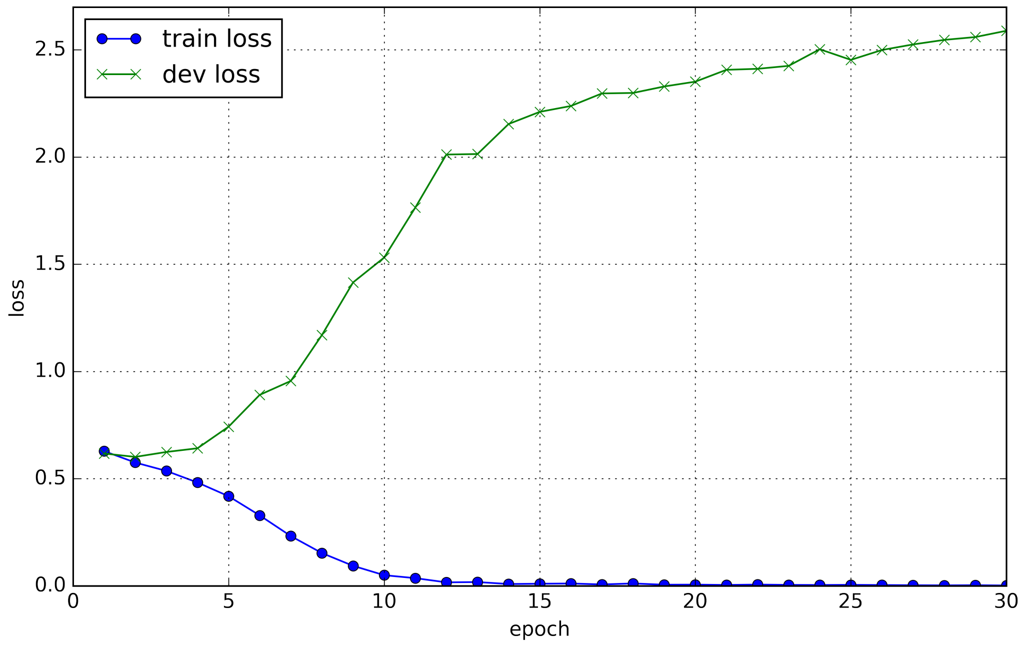


Figure 5.8: Train vs. Dev cross-entropy loss

Results and Analysis

Similarly to other previously evaluated models that were trained only on the MSRP corpus, the classification is done almost random. As can been seen in Figure5.8 and Figure5.9 the features learned during training are irrelevant to the dev dataset. We assume it is due to features like numbers of specific combination of words that happen in the training set that separate the paraphrase classes but are not present in the dev/test set, for instance, certain numbers, named entities, topic specific words, etc.

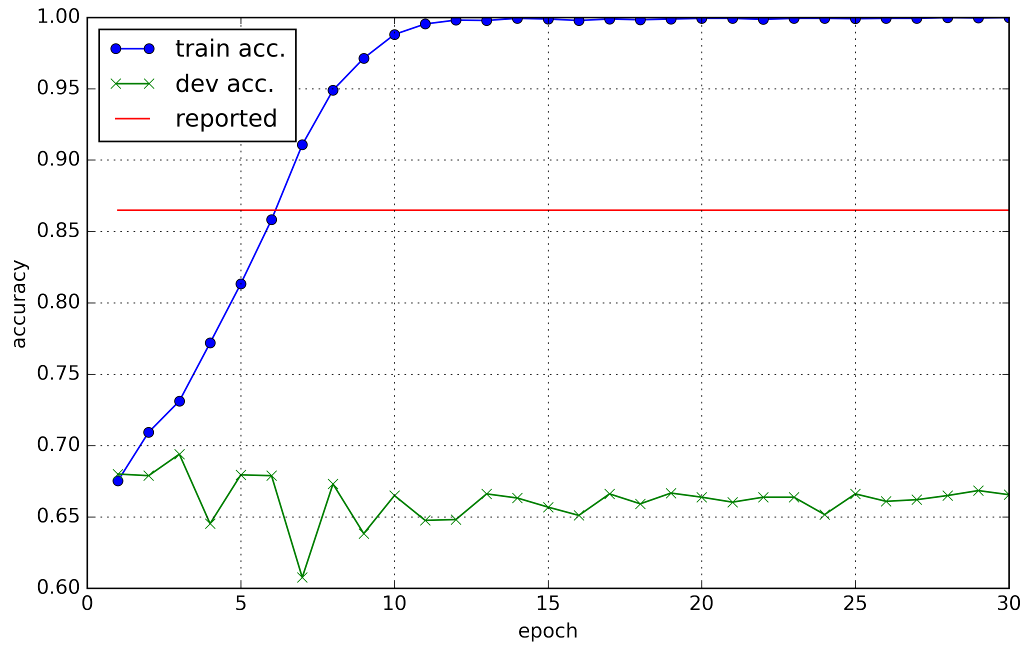


Figure 5.9: Train vs. Dev accuracy

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we addressed the plagiarism detection tasks focused on paraphrase identification. Our strategy to achieve this objective was treating both tasks of plagiarism detection and paraphrase identification, independently.

We proposed new adaptations to our previous state-of-the-art model for plagiarism detection to handle a variety of obfuscation types. We conducted a series of experiments to evaluate the impact of each proposed method outperforming our existing work. We also designed a novel genetic-based model for parameter tuning using non-binary representation where each individual is a possible configuration of our system. With this model we were able to find an optimal set of parameters for our plagiarism detection model with handling the huge search space. We improved the performance further, specifically in the summary obfuscation type.

In the paraphrase identification task we studied two lines of research. First, we worked with knowledge-base approaches using the WordNet ontology. In this regard, we proposed several experimental settings testing combinations of four group of techniques: preprocessing, feature extraction, WordNet related processes, and sentence similarity models. We were able to replicate and outperform methods that used similar techniques but lacked a full description.

The last part of this thesis focused on distributional approaches to paraphrase identification. In that section we made evident the need for more detailed approaches, better training of the models, reproducibility. We also made publicly available our three implementations of the studied models. All of them, are programmed in the same language and using one of the most popular deep learning frameworks (Keras). In one of the cases we decreased the running time 200 times. In another case, our implementation is the first public available one. Also, our systems are the only ones that are designed to be tested on the MSRP dataset without doing any modification.

6.2 Future Work

One of the directions of future work is integrating paraphrase identification into a plagiarism detection system and use our proposed genetic algorithm to optimize the hyper-parameters of the deep learning model. Regarding paraphrase identification, it is relevant to train the state-of-the-art models on larger datasets and apply transfer learning to the specific task at hand. Another possible direction to explore is combining neural networks models with knowledge bases methods to reduce the need of comprehensive datasets that cover all possible scenario.

We will also evaluate combinations of proposed models and study in detail their classification errors. It is important to analyze the impact and constrains introduced by implementing deep learning models using different tools, like padding when using Keras on mini-batches of variable sizes. Another future work needed, although not a scientific contribution, is implementing more models of the state-of-the-art and making them available as open source code.

Bibliography

[1] Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A.: Semeval-2012 task 6: A pilot on semantic textual similarity. In: *SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012). pp. 385–393. Association for Computational Linguistics, Montréal, Canada (7-8 June 2012)

[2] Bach, N.X., Nguyen, M.L., Shimazu, A.: Exploiting discourse information to identify paraphrases. Expert Syst. Appl. 41(6), 2832–2841 (2014)

[3] Bannard, C., Callison-Burch, C.: Paraphrasing with bilingual parallel corpora. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05). pp. 597–604. Association for Computational Linguistics, Ann Arbor, Michigan (June 2005)

[4] Bär, D., Zesch, T., Gurevych, I.: Text reuse detection using a composition of text similarity measures. In: Kay, M., Boitet, C. (eds.) COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8–15 December 2012, Mumbai, India. pp. 167–184. Indian Institute of Technology Bombay (2012)

[5] Barrón-Cedeño, A., Vila, M., Martí, M.A., Rosso, P.: Plagiarism meets paraphrasing: Insights for the next generation in automatic plagiarism detection. Computational Linguistics 39(4), 917–947 (2013)

[6] Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. Journal of Machine Learning Research 3, 1137–1155 (March 2003)

[7] Berant, J., Liang, P.: Semantic parsing via paraphrasing. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1415–1425. Association for Computational Linguistics, Baltimore, Maryland (June 2014)

[8] Bhagat, R., Hovy, E.H.: What is a paraphrase? Computational Linguistics 39(3), 463–472 (2013)

[9] Blacoe, W., Lapata, M.: A comparison of vector-based representations for semantic composition. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. pp. 546–556. Association for Computational Linguistics, Jeju Island, Korea (July 2012)

[10] Bouarara, H.A., Rahmani, A., Hamou, R.M., Amine, A.: Machine learning tool and meta-heuristic based on genetic algorithms for plagiarism detection over mail service. In: Computer and Information Science (ICIS), 2014 IEEE/ACIS 13th International Conference on. pp. 157–162. IEEE (2014)

[11] Cheng, J., Kartsaklis, D.: Syntax-aware multi-sense word embeddings for deep compositional models of meaning. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1531–1542. Association for Computational Linguistics, Lisbon, Portugal (September 2015)

[12] Cheung, M.L.L.: Merging corpus linguistics and collaborative knowledge construction. Ph.D. thesis, University of Birmingham, Birmingham, United Kingdom (2009)

[13] Chomsky, N.: Syntactic Structures. Mouton, The Hague (1957)

[14] Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning. pp. 160–167. ACM, Helsinki, Finland (June 2008)

[15] Dolan, B., Quirk, C., Brockett, C.: Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In: Proceedings of the 20th International Conference on Computational Linguis-

tics. pp. 350–356. Association for Computational Linguistics, Geneva, Switzerland (August 2004)

[16] lester Faigley, Witte, S.: Analyzing revision. College Composition and Communication 4(32), 400–414 (1981)

[17] Fernando, S., Stevenson, M.: A semantic approach to paraphrase identification. In: Proceedings of the 11th Annual Research Colloquium of the UK Special-interest group for Computational Lingusitics. pp. 45–52. None, Oxford, England (Lalala 2008)

[18] Forner, P., Navigli, R., Tufis, D., Ferro, N. (eds.): Working Notes for CLEF 2013 Conference, Valencia, Spain, September 23–26, 2013, CEUR Workshop Proceedings, vol. 1179. CEUR-WS.org (2013)

[19] Ganitkevitch, J., Callison-Burch, C.: The multilingual paraphrase database. In: Proceedings of the 9th edition of the Language Resources and Evaluation Conference. pp. 4276–4283. European Language Resources Association, Reykjavik, Iceland (May 2014)

[20] Ganitkevitch, J., Van Durme, B., Callison-Burch, C.: Ppdb: The paraphrase database. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 758–764. Association for Computational Linguistics, Atlanta, Georgia, USA (June 2013)

[21] Gillam, L.: Guess again and see if they line up: Surrey's runs at plagiarism detection notebook for PAN at CLEF 2013. In: Forner et al. [18]

[22] Gollub, T., Potthast, M., Beyer, A., Busse, M., Pardo, F.M.R., Rosso, P., Stamatatos, E., Stein, B.: Recent trends in digital text forensics and its evaluation - plagiarism detection, author identification, and author profiling. In: Forner, P., Müller, H., Paredes, R., Rosso, P., Stein, B. (eds.) Information Access Evaluation. Multilinguality, Multimodality, and Visualization - 4th International Conference of the CLEF Initiative, CLEF 2013, Valencia, Spain, September 23–26, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8138, pp. 282–302. Springer (2013)

[23] Gollub, T., Stein, B., Burrows, S.: Ousting Ivory Tower research: Towards a web framework for providing experiments as a service. In: Hersh,

B., Callan, J., Maarek, Y., Sanderson, M. (eds.) 35th International ACM Conference on Research and Development in Information Retrieval (SIGIR 12). pp. 1125–1126. ACM (Aug 2012)

[24] Gülich, E.: Conversational techniques used in transferring knowledge between medical experts and non-experts. Discourse Studies 5(2), 235–263 (2003)

[25] Harris, Z.S.: Co-occurrence and transformation in linguistic structure. In: Hiż, H. (ed.) Papers on Syntax, pp. 143–210. Springer Netherlands, Dordrecht, Netherlands (1981)

[26] He, H., Gimpel, K., Lin, J.J.: Multi-perspective sentence similarity modeling with convolutional neural networks. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1576–1586. Association for Computational Linguistics, Lisbon, Portugal (September 2015)

[27] He, H., Wieting, J., Gimpel, K., Rao, J., Lin, J.: Umd-ttic-uw at semeval-2016 task 1: Attention-based multi-perspective convolutional neural networks for textual similarity measurement. In: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016). pp. 1103–1108. Association for Computational Linguistics, San Diego, California (June 2016)

[28] Ji, Y., Eisenstein, J.: Discriminative improvements to distributional sentence similarity. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. pp. 891–896. Asian Federation of Natural Language Processing, Seattle, Washington, USA (October 2013)

[29] Jiang, J.J., Conrath, D.W.: Semantic similarity based on corpus statistics and lexical taxonomy. In: Proceedings of the 10th Research on Computational Linguistics International Conference. pp. 19–33. Association for Computational Linguistics and Chinese Language Processing, Taipei, Taiwan (August 1997)

[30] Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. pp. 655–665.

Association for Computational Linguistics, Baltimore, Maryland, USA (June 2014)

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

[32] Kong, L., Qi, H., Du, C., Wang, M., Han, Z.: Approaches for source retrieval and text alignment of plagiarism detection notebook for PAN at CLEF 2013. In: Forner et al. [18]

[33] Küppers, R., Conrad, S.: A set-based approach to plagiarism detection. In: Forner, P., Karlgren, J., Womser-Hacker, C. (eds.) CLEF 2012 Evaluation Labs and Workshop, Online Working Notes, Rome, Italy, September 17–20, 2012. CEUR Workshop Proceedings, vol. 1178. CEUR-WS.org (2012)

[34] Lange, R.C., Mancoridis, S.: Using code metric histograms and genetic algorithms to perform author identification for software forensics. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. pp. 2082–2089. ACM (2007)

[35] Leacock, C., Chodorow, M.: Combining local context and wordnet sense similarity for word sense identification. WordNet, An Electronic Lexical Database (1998)

[36] Ledeneva, Y.N.: Automatic Language–Independent Detection of Multiword Descriptions for Text Summarization. Ph.D. thesis, Centro de Investigación en Computación, Mexico City, Mexico (2008)

[37] Levin, B.: English Verb Classes and Alternations: A Preliminary Investigation. University of Chicago Press (1993)

[38] Lin, D.: An information-theoretic definition of similarity. In: Proceedings of the Fifteenth International Conference on Machine Learning. pp. 296–304. Morgan Kaufmann, Madison, Wisconsin, USA (July 1998)

[39] Lovins, J.B.: Development of a stemming algorithm. Mechanical Translation and Computational Linguistics 11, 22–31 (1968)

[40] Majumder, N., Poria, S., Gelbukh, A., Cambria, E.: Deep learning-based document modeling for personality detection from text. IEEE Intelligent Systems 32(2), 74–79 (2017)

[41] Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA, USA (1999)

[42] Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., Zamparelli, R.: Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014). pp. 1–8. Association for Computational Linguistics and Dublin City University, Dublin, Ireland (August 2014)

[43] Mashhadirajab, F., Shamsfard, M.: A text alignment algorithm based on prediction of obfuscation types using SVM neural network. In: Majumder, P., Mitra, M., Mehta, P., Sankhavara, J., Ghosh, K. (eds.) Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation, Kolkata, India, December 7-10, 2016. CEUR Workshop Proceedings, vol. 1737, pp. 167–171. CEUR-WS.org (2016)

[44] Maurer, H., Kappe, F., Zaka, B.: Plagiarism – A survey. Journal of Universal Computer Science 12(8), 1050–1084 (Aug 2006)

[45] Mehdizadeh Seraj, R., Siahbani, M., Sarkar, A.: Improving statistical machine translation with a multilingual paraphrase database. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1379–1390. Association for Computational Linguistics, Lisbon, Portugal (September 2015)

[46] Mel'čuk, I.A.: Paraphrase et lexique: la théorie sens-texte et le dictionnaire explicatif et combinatoire. In: Dictionnaire Explicatif et Combinatoire du Français Contemporain. Recherches Lexico-sémantiques III, pp. 9–58. Les Presses de l'Université de Montréal, Montréal, Canada (1992)

[47] Mihalcea, R., Corley, C., Strapparava, C.: Corpus-based and knowledge-based measures of text semantic similarity. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence and the

Eighteenth Innovative Applications of Artificial Intelligence Conference. pp. 775–780. AAAI Press, Boston, Massachusetts, USA (July 2006)

[48] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013), `http://arxiv.org/abs/1301.3781`

[49] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26, pp. 3111–3119. Curran Associates, Inc. (2013)

[50] Milićević, J.: La Paraphrase. Modélisation de la Paraphrase Langagière. Peter Lang, Bern (2007)

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

[52] Mitchell, J., Lapata, M.: Composition in distributional models of semantics. Cognitive Science 34(8), 1388–1429 (2010)

[53] Mitchell, M.: An introduction to genetic algorithms. MIT Press (1998)

[54] Neelakantan, A., Shankar, J., Passos, A., McCallum, A.: Efficient non-parametric estimation of multiple embeddings per word in vector space. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1059–1069. Association for Computational Linguistics, Doha, Qatar (October 2014)

[55] Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., Song, X., Ward, R.: Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. IEEE/ACM Trans. Audio, Speech and Lang. Proc. 24(4), 694–707 (April 2016)

[56] Palkovskii, Y., Belov, A.: Using hybrid similarity methods for plagiarism detection notebook for PAN at CLEF 2013. In: Forner et al. [18]

[57] Patwardhan, S., Banerjee, S., Pedersen, T.: Using measures of semantic relatedness for word sense disambiguation. In: Computational Linguistics and Intelligent Text Processing, Proceedings of the 4th edition of CICLing Conference. pp. 241–257. Springer, Mexico City, Mexico (February 2003)

[58] Pavlick, E., Rastogi, P., Ganitkevitch, J., Van Durme, B., Callison-Burch, C.: Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers). pp. 425–430. Association for Computational Linguistics, Beijing, China (July 2015)

[59] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. pp. 1532–1543. Association for Computational Linguistics, Doha, Qatar (October 2014)

[60] Porter, M.F.: An algorithm for suffix stripping. In: Sparck Jones, K., Willett, P. (eds.) Readings in Information Retrieval, pp. 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1997)

[61] Potthast, M., Barrón-Cedeño, A., Eiselt, A., Stein, B., Rosso, P.: Overview of the 2nd international competition on plagiarism detection. In: Braschler, M., Harman, D., Pianta, E. (eds.) CLEF 2010 LABs and Workshops, Notebook Papers, 22–23 September 2010, Padua, Italy. CEUR Workshop Proceedings, vol. 1176. CEUR-WS.org (2010)

[62] Potthast, M., Hagen, M., Beyer, A., Busse, M., Tippmann, M., Rosso, P., Stein, B.: Overview of the 6th International Competition on Plagiarism Detection. In: Cappellato, L., Ferro, N., Halvey, M., Kraaij, W. (eds.) Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15–18, 2014. CEUR Workshop Proceedings, vol. 1180, pp. 845–876. CEUR-WS.org (2014)

[63] Potthast, M., Hagen, M., Gollub, T., Tippmann, M., Kiesel, J., Rosso, P., Stamatatos, E., Stein, B.: Overview of the 5th International Competition on Plagiarism Detection. In: Forner et al. [18]

[64] Potthast, M., Stein, B., Barrón-Cedeño, A., Rosso, P.: An evaluation framework for plagiarism detection. In: Huang, C., Jurafsky, D. (eds.) COLING 2010, 23rd International Conference on Computational Linguistics, Posters Volume, 23-27 August 2010, Beijing, China. pp. 997–1005. Chinese Information Processing Society of China (2010)

[65] Potthast, M., Stein, B., Eiselt, A., Barrón-Cedeño, A., Rosso, P.: Overview of the 1st international competition on plagiarism detection. In: In: SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09), CEUR-WS.org. pp. 1–9 (2009)

[66] Quirk, C., Brockett, C., Dolan, W.B.: Monolingual machine translation for paraphrase generation. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain. pp. 142–149. ACL (2004)

[67] Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010)

[68] Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. pp. 448–453. Morgan Kaufmann, Montreal, Canada (August 1995)

[69] Rodríguez Torrejón, D.A., Martín Ramos, J.M.: Text alignment module in CoReMo 2.1 plagiarism detector notebook for PAN at CLEF 2013. In: Forner et al. [18]

[70] Rodríguez Torrejón, D.A., Martín Ramos, J.M.: Coremo 2.3 plagiarism detector text alignment module. In: CLEF (Working Notes). pp. 997–1003 (2014)

[71] Sanchez-Perez, M.A., Gelbukh, A., Sidorov, G.: Adaptive algorithm for plagiarism detection: The best-performing approach at PAN 2014 text alignment competition. In: Mothe, J., Savoy, J., Kamps, J., Pinel-Sauvagnat, K., Jones, G.J.F., SanJuan, E., Cappellato, L., Ferro, N. (eds.) Experimental IR Meets Multilinguality, Multimodality, and Interaction - 6th International Conference of the CLEF Association, CLEF 2015, Toulouse, France, September 8–11, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9283, pp. 402–413. Springer (2015)

[72] Sánchez-Pérez, M.A., Gelbukh, A., Sidorov, G.: Dynamically adjustable approach through obfuscation type recognition. In: Cappellato, L., Ferro, N., Jones, G.J.F., SanJuan, E. (eds.) Working Notes of CLEF 2015—Conference and Labs of the Evaluation forum, Toulouse, France, September 8–11, 2015. CEUR Workshop Proceedings, vol. 1391. CEUR-WS.org (2015), `http://ceur-ws.org/Vol-1391`

[73] Sanchez-Perez, M.A., Sidorov, G., Gelbukh, A.: The winning approach to text alignment for text reuse detection at PAN 2014. In: Working Notes for CLEF 2014 Conference. pp. 1004–1011. CEUR-WS.org, Sheffield, UK (September 2014)

[74] dos Santos, C., Gatti, M.: Deep convolutional neural networks for sentiment analysis of short texts. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. pp. 69–78. Dublin City University and Association for Computational Linguistics, Dublin, Ireland (August 2014)

[75] Shen, Y., Chen, J., Huang, X.: Bidirectional long short-term memory with gated relevance network for paraphrase identification. In: Natural Language Understanding and Intelligent Applications, pp. 39–50. Springer International Publishing (2016)

[76] Shrestha, P., Solorio, T.: Using a variety of n-grams for the detection of different kinds of plagiarism notebook for PAN at CLEF 2013. In: Forner et al. [18]

[77] Sidorov, G., Gelbukh, A., Gómez-Adorno, H., Pinto, D.: Soft similarity and soft cosine measure: Similarity of features in vector space model. Computación y Sistemas 18(3) (2014)

[78] Sierra, S., Montes-Y-Gómez, M., Solorio, T., González, F.: Convolutional neural networks for author profiling in pan 2017. In: CLEF 2017 Evaluation Labs and Workshop - Working Notes Papers. CEUR-WS.org, Dublin, Ireland (September 2017)

[79] Socher, R., Huang, E.H., Pennin, J., Manning, C.D., Ng, A.Y.: Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In: Proceedings of the 24th International Conference on Neural Information Processing Systems. pp. 801–809. Curran Associates Inc., Granada, Spain (December 2011)

[80] Stamatatos, E.: Plagiarism detection using stopword n-grams. JASIST 62(12), 2512–2527 (2011)

[81] Stein, B., zu Eissen, S.M., Potthast, M.: Strategies for retrieving plagiarized documents. In: Kraaij, W., de Vries, A.P., Clarke, C.L.A., Fuhr, N., Kando, N. (eds.) SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007. pp. 825–826. ACM (2007)

[82] Suchomel, S., Kasprzak, J., Brandejs, M.: Diverse queries and feature type selection for plagiarism discovery notebook for PAN at CLEF 2013. In: Forner et al. [18]

[83] Turian, J., Ratinov, L., Bengio, Y.: Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. pp. 384–394. Association for Computational Linguistics, Uppsala, Sweden (July 2010)

[84] Wang, Z., Mi, H., Ittycheriah, A.: Sentence similarity learning by lexical decomposition and composition. In: Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers. pp. 1340–1349. Association for Computational Linguistics, Osaka, Japan (December 2016)

[85] Wieting, J., Bansal, M., Gimpel, K., Livescu, K.: From paraphrase database to compositional paraphrase model and back. TACL 3, 345–358 (2015)

[86] Wu, Z., Palmer, M.: Verb semantics and lexical selection. In: Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics. pp. 133–138. Association for Computational Linguistics, Las Cruces, New Mexico, USA (June 1994)

[87] Yin, W., Schütze, H., Xiang, B., Zhou, B.: ABCNN: Attention-based convolutional neural network for modeling sentence pairs. arXiv preprint arXiv:1512.05193 (2015)

[88] Zhang, Y., Wallace, B.C.: A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. CoRR abs/1510.03820 (2015)