FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Using named entity recognition for relevance detection in social network messages

Filipe Daniel da Gama Batista

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Álvaro Pedro de Barros Borges Reis Figueira

July 23, 2017

Using named entity recognition for relevance detection in social network messages

Filipe Daniel da Gama Batista

Mestrado Integrado em Engenharia Informática e Computação

July 23, 2017

Abstract

The continuous growth of social networks in the past decade has led to massive amounts of information being generated on a daily-basis. While a lot of this information is merely personal or simply irrelevant to a general audience, relevant news being transmitted through social networks is an increasingly common phenomenon, and therefore detecting such news automatically has become a field of interest and active research.

The contribution of the present thesis consisted in studying the importance of named entities in the task of relevance detection. With that in mind, the goal of this work was twofold: 1) to implement or find the best named entity recognition tools for social media texts, and 2) to analyze the importance of extracted entities from posts as features for relevance detection with machine learning.

There are already well-known named entity recognition tools, however, most state-of-the-art tools for named entity recognition show significant decrease of performance when tested on social media texts, in comparison to news media texts. This is mainly due to the informal character of social media texts: the absence of context, the lack of proper punctuation, wrong capitalization, the use of characters to represent *emoticons*, spelling errors and even the use of different languages in the same text. To address these problems, four different state-of-the-art toolkits — Stanford NLP, GATE with TwitIE, Twitter NLP tools and OpenNLP — were tested on social media datasets. In addition, we tried to understand how differently these toolkits predicted Named Entities, in terms of their precision and recall for three different entity types (PERSON, LOCATION, ORGANIZATION), and how they could complement each other in this task in order to achieve a combined performance superior to each individual one, creating an Ensemble of Toolkits.

The developed Ensemble was then used to extract entities and different features were generated: the number of persons, locations and organizations mentioned in a post, statistics retrieved from *The Guardian*'s open API, and finally word embeddings. Multiple machine learning models were then trained on a relevance-labelled dataset of tweets. The obtained performances of different combinations of selected features, ML algorithms, hyperparameters, and datasets, were analyzed.

Our results on two publicly available datasets from the workshops WNUT-2015 and #MSM-2013 showed that the Ensemble of Toolkits was able to improve the recognition of specific entity types - up to 10.62% for the entity type PERSON, 1.97% for the type LOCATION and 1.31% for the type ORGANIZATION, depending on the dataset and the criteria used for the voting. Our results also showed improvements of 3.76% and 1.69%, in each dataset respectively, on the average performance of the three entity types.

The relevance analysis showed that Named Entities can indeed be useful for relevance detection, proving to be useful not only when used alone, having achieved up to 73.4% of AUC, but also helpful when combined with other features such as word embeddings, having achieved a maximum AUC of 92.7% in that case, a 1.4% improvement over word embeddings alone.

Resumo

O crescimento contínuo das redes sociais ao longo da última década levou a que quantidades massivas de informação sejam geradas diariamente. Enquanto grande parte desta informação é de índole pessoal ou simplesmente sem interesse para a população em geral, tem-se por outro lado vindo a testemunhar cada vez mais a propagação de notícias importantes através de redes sociais. Como tal, a deteção automática destas notícias é uma temática cada vez mais investigada.

Esta tese foca-se no estudo da relação entre entidades mencionadas numa publicação de rede social e a respetiva relevância jornalística dessa mesma publicação. Nesse sentido, este trabalho foi dividido em dois grandes objetivos: 1) implementar ou encontrar o melhor sistema de reconhecimento de entidades mecionadas (REM) para textos de redes sociais, e 2) analisar a importância de entidades extraídas de publicações como atributos para deteção de relevância com aprendizagem computacional.

Apesar de existirem diversas ferramentas para extração de entidades, a maioria apresenta uma perda significativa de desempenho quando testada em textos de redes sociais. Isto deve-se essencialmente à informalidade característica deste tipo de textos, traduzida pela ausência de contexto, pontuação desadequada, capitalização errada, representação de *emoticons* com recurso a caracteres, erros gramaticais ou lexicais e até mesmo utilização de diferentes línguas no mesmo texto. Face a estes problemas, quatro ferramentas de reconhecimento de entidades — Stanford NLP, Gate com TwitIE, Twitter NLP tools e OpenNLP — foram testadas em "datasets" de redes sociais. Para além disso, tentamos compreender quão diferentemente é que estas ferramentas se comportavam, em termos de Precisão e *Recall* para 3 tipos de entidades (PESSOA, LOCAL e ORGANIZAÇÃO), e de que forma estas ferramentas se poderiam complementar de forma a obter um desempenho superior, criando assim um *Ensemble* de ferramentas de REM. O Ensemble desenvolvido foi então utilizado para extrair entidades, e diferentes atributos foram gerados: o número de pessoas, locais e organizações mencionados numa publicação, estatísticas obtidas a partir da API pública do jornal *The Guardian*, e finalmente *word embeddings*. Vários modelos de aprendizagem foram treinados num dataset de *tweets* manualmente anotados. Os resultados obtidos das diferentes combinações de atributos, algoritmos, *hyperparameters* e datasets foram analisados.

Os nossos resultados em dois datasets públicos, das conferências WNUT-2015 e #MSM2013, mostraram que utilizar o Ensemble de ferramentas de NER melhorou o reconhecimento de tipos de entidade específicos - até 10.62% para PESSOA, 1.97% para LOCAL e 1.31% para ORGANIZAÇÃO, dependendo do dataset e do critério de voto utilizado. Os resultados motraram também melhorias de 3.76% e 1.69% na média geral dos três tipos de entididades, em cada um dos datasets mencionados, respectivamente.

A análise de relevância mostrou que entidades mencionadas numa publicação podem de facto ser úteis na deteção da sua relevância, não apenas quando usadas isoladamente, tendo alcançado até 73.4% de AUC nesse caso, mas também úteis quando combinadas com outros atributos como *word embeddings*, tendo alcançado nesse caso um máximo de 92.7%, uma melhoria de 1.4% em relação a usar exclusivamente *word embeddings*.

Acknowledgements

First, I would like to thank my supervisor Álvaro Figueira, for his help and guidance along the course of this thesis.

I would also like to thank the whole team of project REMINDS, especially my friends Filipe Miranda and Nuno Guimarães for their constant availability and helpful suggestions. I would also like to extend my appreciation to the “ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness)" and “FCT (Portuguese Foundation for Science and Technology)", for financing project REMINDS and my research grant.

Last but not least, I would like to express my deepest gratitude for everyone that supported me throughout this journey, including my family, girlfriend and closest friends.

Filipe Batista

"The important thing is not to stop questioning.
Curiosity has its own reason for existing."

Albert Einstein

Contents

List of Figures

List of Tables

Abbreviations

NLP	Natural language processing
NER	Named entity recognition
POS	Parts-of-speech
SVM	Support vector machines
CoNLL	Conference on Computational Natural Language Learning
CRF	Conditional Random Field
REMINDS	Relevance Mining Detection System
ML	Machine Learning
API	Application Programming Interface
TP	True positive
TN	True negative
FP	False positive
FN	False negative
SFS	Sequential Forward Selection
SBS	Sequential Backward Selection

Chapter 1

Introduction

This chapter will present the context of this thesis, not only in terms of its scientific domain but also in terms of the research project this thesis finds itself integrated in. The motivations and goals of this work will be explained, along with problems it aims to solve.

This introduction will also provide a brief summary of each of the following chapters.

1.1 Context

Everyday millions of people worldwide use social networks to communicate and share their opinions about virtually any topic. Well-known social networks such as Facebook and Twitter generate and store massive amounts of data every second, constantly increasing potentially useful data available for study. With so much data being stored everyday emerges the obvious opportunity to analyze such data and learn from it.

On one hand, people began to willingly leave their footprint behind, by not only commenting and leaving their opinions on the Internet but also by creating and filling in their own online profiles. Therefore, most of this data could be considered personal and irrelevant to a general audience.

On the other hand, the phenomena of relevant information being transmitted through social networks before being reported by the traditional media is becoming increasingly common [KLPM10]. In fact, social networks have enabled anyone with access to the Internet to report incidents around them in real time, creating a global “word of mouth" effect, as witnessed in multiple situations such as the recent Paris and London terror attacks.

The ability to detect and gather such information from multiple and scattered sources within the massive stream of posts generated by these social networks is therefore very useful for news agencies [BG13], which need to review and structure the information as soon as possible in order to deliver it as “news" to the public. This task, however, is very hard and presents great challenges becoming, as a result, a field of active research in recent years [MCG15, GAPGC^{+}13].

1.2 Project

The work of this thesis is integrated in a international research project, REMINDS, which stands for RElevance MINing Detection System.

This project is supported by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT (Portuguese Foundation for Science and Technology).

The goal of REMINDS is to build a system capable of automatically detecting relevant information (in a news-worthy sense of relevance) in social media, by using machine learning to develop predictive data mining models.

To achieve this purpose, different algorithms have been be trained and tested on different manually labeled datasets, with a number of extracted features, in an attempt to obtain the best model able to predict the relevance of a social media post.

1.3 Motivations and Goals

While several studies have been conducted to address the problem of relevance detection [MCG15, GAPGC+13], there is still room for research.

The main motivation of the present thesis is to contribute to such research, by analyzing the potential usefulness of named entities mentioned in a post for assessing the relevance of the corresponding post.

A named entity is a “sequence of words that designate some real-world entity" [AZ12], for example, the words “New York" should be identified and classified with the type LOCATION, while the words “Barack Obama" should be classified with the type PERSON. The process of identifying named entities in plain text is called Named Entity Recognition (NER), a sub-field of Information Extraction.

That being said, the main goal of this thesis is to try multiple features based on named entities mined from the text of the posts, such as the number of persons, locations and organizations mentioned in a post, together with features previously used in the project, and analyze how these features improve the system.

However, mining text data on social networks presents great challenges [RCE+11]. The main challenge consists in the difficulty of dealing with its non-standard vocabulary: the absence of context, the lack of proper punctuation, wrong capitalization, the use of characters to represent emoticons, spelling errors and even the use of different languages in the same text, are some examples of common problems which have been shown to significantly decrease named entity recognition performance, even on reputable state-of-the-art tools.

For that reason, the intermediate goal of this thesis is to study different state-of-the-art tools for NER — namely Stanford NER, OpenNLP, Twitter NLP tools and GATE with TwitIE — in the context of social media.

In an attempt to attenuate the limitations imposed by the characteristics of this type of text, it is our goal to combine these tools to achieve the best possible performance in the task of entity recognition.

This will allow a more efficient entity extraction on posts from Facebook or Twitter, which will serve as the basis for our main purpose of studying relevance detection.

In summary, the goals of this thesis are as follows:

1. Compare different state-of-the-art tools for Named Entity Recognition in social network messages
2. Analyze how differently they perform in terms of the core entity types: PERSON, LOCATION, ORGANIZATION
3. Study the possible complementary nature of such toolkits, by using rules and machine learning to combine their outputs
4. Create an automatized ensemble of such NER tools with enhanced performance and use it for extracting entities from the relevance-labeled dataset
5. Perform feature extraction from the entities obtained in the previous goal
6. Study the importance of named entities for the task of relevance detection

1.4 Thesis structure

This thesis is structured as follows. In chapter 2, a literature review on Named Entity Recognition will be presented, including its context on Natural Language Processing, the methodologies normally used to perform this task, and the differences and difficulties encountered when performing Named Entity Recognition on social media texts versus formal texts. Chapter 3 will provide a brief literature review on feature analysis for classification systems, including feature engineering, selection and generation techniques. Chapter 4 will be focused on the utilization of toolkits for the extraction of named entities, as well as the results of a developed Ensemble of those NER tools. In chapter 5, we will cover the process of features extraction based on named entities, and the importance of each feature in different machine learning models for relevance detection. Finally, in chapter 6, we will summarize the results of the whole work and present our final conclusions.

Chapter 2

Named Entity Recognition: Literature review

This chapter presents the state of the art of Named Entity Recognition on different types of text, by reviewing related works and studies found in literature.

By the end of this section it should be clear what the current state of named entity recognition is, not only on formal texts but especially on social media texts, and what measures can be implemented in order to tackle some of the challenges of these types of texts.

This chapter will also cover different tools chosen to be experimented in this project, and provide a brief summary about each of these tools.

2.1 Introduction

Information extraction is the branch of text mining that aims to gather information from unstructured data. Named entity recognition is a sub-field of information extraction: its task is to structure data by finding names of entities such as people, organizations and locations within some text, and tag them with the respective predefined “label".

For example, given the unstructured sentence “In 2016, Donald Trump won the presidency of the United States", it would be expected of a named entity recognizer the following output:

“In **[2016]**$_{\text{time}}$, **[Donald Trump]**$_{\text{person}}$ won the presidency of the **[United States]**$_{\text{location}}$”

2.2 The NLP Pipeline

In order to perform named entity recognition on a text, usually some preprocessing steps need to be taken beforehand. This preprocessing is often split into separate but sequential modules, where the output of one module is used as input by the next module, following what is called a pipeline design.

There is no standard structure for NLP pipelines, since multiple variations of the same can be found in literature. However, there are some common steps among most of the pipelines proposed, including tokenization, POS tagging, chunking and finally NER itself.

2.2.1 Tokenization

The tokenization step consists in dividing the text into multiple tokens. In the current context, tokens are usually words. While tokenization might appear to be a simple task, rules must be defined in order to clarify what a word is. A naive way to do so would be to consider a token every string separated by a whitespace. For instance, given the input string “In 2016, Donald Trump won the presidency of the United States", the corresponding output should be the following list of words: [’In’, ’2016,’,’Donald’, ’Trump’,’won’, ’the’, ’presidency’, ’of’, ’the’, ’United’, ’States’,’.’].

However, simple approaches are usually unable to solve certain ambiguities. For example, the commas and periods in “100.200,10" have a specific purpose and therefore the number should not be split as if it was a sentence [ARS13].

Current state of the art tokenizers such as the Stanford tokenizer use more sophisticated heuristics that allow it to determine when punctuation implies sentence boundaries, when single quotes are parts of words, etc. Recently, a high performance tokenizer was proposed, claiming to be 13.9 times faster than the Stanford while using half the memory [ARS13].

2.2.2 POS tagging

A Part-Of-Speech Tagger (POS Tagger), assigns a part-of-speech to each token previously extracted. Examples of parts-of-speech are nouns, verbs, adjectives, adverbs, and determiners. There are different POS Tagsets available, the most commonly used being the Penn TreeBank [MMS93], composed of 45 tags.

Using the Stanford POS tagger as a state-of-the-art example, for the input string “This is an example sentence", the output tagged sentence would be “This_DT is_VBZ an_DT example_NN sentence_NN ", where DT stands for determiner, VBZ for Verb in the 3rd person singular present, and NN for Noun, singular.

Stanford NLP pipeline offers two different models for POS tagging, namely a bidirectional model and L3W (Left 3 words) model, the latter being the preferred tagger to use in practice. These models achieved respective accuracies of 97.32% and 96.89%, but the LW3 model was 18 times faster than the Bidirectional model [ARS13].

2.2.3 Chunking

Chunking (or shallow parsing), consists in finding short phrases, such as noun-phrases, inside a sentence.

While POS tagging assigns parts-of-speech to each token individually, chunking goes further by grouping different parts-of-speech to give some insight about the structure of the sentence.

Taking the example given by Bird et al. [BKL09], for the input sentence: “We saw the yellow dog", the tagging output should have been “We_PRB saw_VBD the_DT yellow_JJ dog_NN ". Then, this tagged sentence would be fed as input to the chunking module, which would output the following “[We]_NP saw [the yellow dog]_NP".

In this case, “saw" was not grouped in any chunk. The standard representation for chunking is the IOB representation. “I" stands for an inside chunk, “O" for an outside chunk and “B" for the beginning of a chunk. Using this notation for the previous example, the output of the chunker would be: “[We]_B-NP [saw]_O [the]_B-NP [yellow]_I-NP [dog]_I-NP". [BKL09]

Chunking can be seen as a middle ground between full parsing and POS tagging. The idea behind chunking is that full parsing is computationally expensive and not always necessary. Besides, parsing adapts badly to new domains [BKL09], such as Twitter.

Chunkers can be regular expression-based, using only part-of-speech tags to decide the chunks. However using part-of-speech tags only and ignoring the content of the the words might not be sufficient. To solve this problem, a classifier-based chunker could be used [BKL09].

2.2.4 NER

The output from chunking will provide useful features for named entity recognition. For example, named entities are essentially definite noun phrases [BKL09], which were previously labeled by the chunking module.

The NLP pipeline usually does not end in entity recognition. It could proceed to other tasks after named entity recognition, for example, relation extraction. For the scope of this work, however, further NLP tasks will not be discussed.

The entity recognition module of the pipeline will be comprehensively explained in the next section.

2.3 Methodologies

To perform Named Entity Recognition, different approaches were proposed over time. Early systems started by implementing dictionaries, relying on a list of named entities [WA86]. Later, rule-based approaches were introduced to complement dictionary-based approaches and solve problems such as proper name recognition [SN14]. Nowadays, many state of the art entity recognizers use supervised machine learning to reduce the work of manually of defining rules, by automatizing the process.

2.3.1 Dictionary-based Approach

This approach makes use of a dictionary (also commonly referred to as gazetteer or lexicon) of named entities. A dictionary is no more than a list of words already labeled with a named entity.

For every word in a text, the algorithm simply checks its corresponding category in the dictionary, and assigns that category to the word.

The obvious problem with this approach is the very large number of words existent and the impracticability of manually defining the correct entity of each one, and also the ambiguity of some words whose entity type might be context-dependent. Most NER systems also need frequent updates, which are not practical with this approach.

Dictionary-based approaches can be useful, however, in conjunction with other approaches or in specific contexts where we are only interested in identifying entities of a small and not very dynamic dictionary [Ker16].

2.3.2 Rule-based Approach

Rule-based approaches have also been widely used in Named Entity Recognition.

These approaches rely on a set of rules, which are no more than “if-then" statements: if a pattern, which can be defined by a regular expression for example, is detected in the text, then a predefined corresponding action is triggered.

Let us follow the example from Mining Text Data (Charu C. Aggarwal,ChengXiang Zhai,2011) [AZ12]:

> “to label any sequence of tokens of the form “Mr. X" where X is a capitalized word as a person entity, the following rule can be defined:
>
> **(token = “Mr." orthography type = FirstCap) -> person name**
>
> The left hand side is a regular expression that matches any sequence of two tokens where the first token is “Mr." and the second token has the orthography type FirstCap. The right hand side indicates that the matched token sequence should be labeled as a person name.”

To prevent cases where multiple rules happen to be matched in a given string, a protocol should be defined to handle these conflicts appropriately. A simple way to handle conflicts is to order the rules by their priority, so that only the “stronger" rule is applied, for example. [AZ12]

These rules can either be defined manually or learned automatically (explained in the next subsection in further detail). The first approach is very demanding, and therefore expensive. The latter approach transfers most of the workload to the computer, although it still needs a training set with manually labeled named entities.

2.3.3 Machine Learning Approaches

Machine learning approaches have also been widely used for Named Entity Recognition. They are more versatile than rule-based approaches, and require less hand-work.

NER can be seen as multi-class classification problem. In this case, a dataset using some features will be used to train a model.

2.3.3.1 Algorithms

One of the most commonly used ML approaches for named entity recognition is Conditional Random Fields (CRF), a special case of Markov Random Fields [AZ12]. The widely known Stanford Named Entity Recognizer implements a CRF [FGM05].

Notwithstanding, many other ML algorithms can be used for NER. In fact, sixteen systems have participated in the CoNLL-2003 shared task using multiple different machine learning approaches, including Maximum Entropy Models (MEM), Hidden Markov Models (HMM), Support Vector Machines (SVM), among others. Some of the participants of the ConLL-2003 used combinations of previously mentioned ML algorithms, with good results. [TKSDM03]

Ensemble learning for named entity recognition has also been evaluated in [SN14], where the authors compared several ML algorithms with this approach. This work concluded that NER based on ensemble learning can achiever higher F1-scores than most state-of-the-art algorithms (Stanford NER, Illionois,Balie, OpenNLP) and higher F1-scores than simple voting methods (studied for example by Wu et al. [WNC03]).

ML approaches can also be used in conjunction with previously explained approaches such as Rule or Dictionary based. In fact, most of the current implementations of Named Entity Recognition rely on a combination of some machine learning algorithm with one or even both of these approaches [Ker16].

2.3.3.2 Features

According to [TKSDM03], choosing good features is not only important but at least as important as choosing the learning approach.

In the ConLL-2003 shared task [TKSDM03] the features used included affix information (n-grams), parts-of-speech, bag-of-Words, global case information, chunk tags, global document information, gazetteers, lexical features, orthographic information and patters, previously predicted named entities, quotation information and trigger words.

Studies have been conducted to study the best features for Named Entity Recognition. Tkachenko et al. [TS12] explored multiple combinations of features and compared their impact on NER performance. The results showed that the set of features used had a significant impact on the performance of NER.

In summary, most state-of-the-art tools rely on machine learning approaches for Named Entity Recognition, due to their versatility and reduced overhead. However, other methods might be helpful in a few cases, such as when dealing with specific small domains.

2.4 Evaluation measures in NLP

Performance in classification systems is measured by comparing the output of a classifier on unseen data with a golden standard - made by human annotators, and assumed as correct. A certain prediction can be either Positive or Negative, and according to the golden standard, that prediction can be True or False.

In the context of information extraction: [GAT]

- True Positive (TP) - corresponds to a correct annotation
- False Positive (FP) - the annotation is present in the output set but not in the golden standard set
- False Negative (FN) - the annotation is present in the golden standard set but not in the output set
- True Negative (TN) - the annotation is absent in both sets. Therefore, it is not considered in IE measures.

There are two ways of counting true positives. The strict way considers “only correct annotations covering exactly the correct span of text" while the leniant way considers that “«partially correct» or «overlap» annotations also count as correct: correct annotation covering part of the correct span of text (shorter, longer, overlapping at either end)" [GAT]. Common metrics to measure performance of classification tasks include:

Accuracy: Calculates the percentage of correct predictions.

$$A = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.1}$$

Precision: Represents the ratio between correctly identified instances of a certain class and all the predictions made for that class.

$$P = \frac{TP}{TP+FP} \tag{2.2}$$

Recall: Represents the ratio between correctly identified instances of a certain class and all the instances of that class. Thus, it tells us how much of the class is being predicted by the classifier.

$$R = \frac{TP}{TP+FN} \tag{2.3}$$

Recall and Precision measures are not sufficient when used independently, meaning that knowing recall without knowing precision, or vice-versa, does not provide enough information about the performance of the system. The most common way to combine Recall and Precision in one single measure is the F-measure.

F-measure: Calculates the harmonic mean of precision and recall. The relative importance (weight) of each component (precision and recall) is controlled by the β parameter (higher values of β mean more weight on recall) [CFL13].

$$F_\beta = \frac{(\beta^2+1)\times P\times R}{\beta * P + R} \tag{2.4}$$

F1 score: Used when both measures have the same importance (β = 1)

$$F_1 = \frac{2\times P\times R}{P+R} \tag{2.5}$$

2.5 Corpora types

2.5.1 Formal text

Most of the initial NER research focused on news articles, scientific articles, books or web pages [MWC05].

The text found in these types of documents is usually formal: writers are generally careful and therefore mistakes or non-standard words are not to be expected. Besides, formal texts are usually meant to be understood by anyone who could possibly read them. For this reason, they are usually clear and avoid incurring in context ambiguities.

This type of text has been studied extensively through different approaches and state-of-the-art systems, such as the Stanford NER, achieving F1 scores of over 92.0% [LZWZ11].

2.5.2 Informal text

Named entity recognition on formal text has been studied for longer than social media text. NER research started growing in early 1990's, while major social networks appeared more than a decade later - Facebook was founded in 2004 and Twitter in 2006.

With huge (and still growing) social networks such as these, generating huge amounts of text data mostly written by the users, the study of NER applied to social media texts has increased greatly over time.

It is important to stress that social media texts differ from previously studied texts in their lack of formality and standardization. These differences have been shown to severely degrade the performance of standard NLP tools [RCE+11], and therefore multiple research studies have been conducted in an attempt to attenuate such losses.

Some of these problems, and proposed approaches in the literature, are listed below:

- **Lexical variations**

  In social media it is very frequent to see lexical variations of words, either misspellings or often just to make texts more compact. For example, “tomorrow" could be written as “tomorow", “2morrow" or even “2m". An usually effective way to deal with these lexical variations is by using brown clustering [RCE+11], which relies on distributions to determine the similarity between words. The intuition behind brown clustering is that similar words appear in similar contexts, i.e. similar words have similar distributions of words that precede and follow them. This method can be applied to misspellings and abbreviations.

- **Non-standard capitalization**

  Capitalization is an important detail in POS tagging and Named entity recognition since it often carries the information of whether a word is a proper noun or a common noun. For example, losing this information might lead a named entity recognizer to understand “apple" as a fruit, when it actually meant “Apple", the technology company.

  In [DMR$^+$15] a comparison of NER performances between capitalized and non-capitalized text was conducted, using the Standord NER, part of the CoNLL dataset (Tjong Kim Sang et al., 2003), and the same dataset with all tokens lowercased. In the first dataset (the original) the authors obtained 89.2 precision and 88.5 recall. In the lowercased dataset, the precision dropped to 81.7 and the recall to 4.1.

  Restoring capitalization (truecasing) is therefore useful for multiple NLP tasks. Strategies to overcome the problem of unreliable capitalization have been proposed in existing literature. A recent article [NBG15] uses an SVM classifier to determine if the capitalization of a given tweet is informative or uninformative. According to [NBG15], this approach performed better than the Stanford Truecaser which implements a discriminative model using a CRF sequence tagger.

- **Wrong punctuation and use of emoticons**

  As previously mentioned, punctuation is very important in NLP tasks, especially for the tokenization part, since it often uses punctuation as one of the criteria to delimit tokens. Many users, however, use punctuation to different purposes, such as to elaborate emoticons: “8<:-)”, “orz” (represents a kneeling man), “<(-’.’-)>”, “(ò_ó)” or “(=^-^=)”. The range of possibilities is so large that it would be nearly impossible to cover all of them in a dictionary. The use of characters to represent emoticons presents a challenge in the tokenization process, as punctuation is very important to determine where two tokens should be separated [LSTO10]. In [LSTO10] a classification-based approach using SVM was tested, and shown to outperform rules manually defined for the purpose of tokenization.

- **Use of multiple languages**

  Most information extraction methods are primarily focused on the English language. However, the presence of other languages in social media texts is far from negligible. In fact, a study has shown that in a 62 million tweet dataset only 51% of the tweets were actually written in English [HCC11]. Most of the algorithms used for English text perform worse when applied to other languages, and are also difficult to adapt to multiple languages [DMR$^+$15].

One approach to deal with this problem is to identify the language being used before analyzing the text, and then choosing the algorithm according to the language. The TwitIE is an example of a toolkit which implements language identification as the first step of the pipeline [BDF+13].

This approach, however, does not solve the problem of multiple languages being used within the same post.

- **Infrequent named entity types and lack of context**

  The fact that tweets are limited to short messages means they often lack sufficient context to disambiguate the types of the entities mentioned in them. Usually, we (humans) can easily understand these mentions because we have some prior knowledge that allows us to understand what the likely context of that post is. For a computer, however, this can be difficult to achieve since more training data is needed. This problem gets even worse due to the fact that many different entity types can be found in Tweets, requiring even more training data [RCE+11]. Ritter et al. addresses this problem with a distant supervision approach applying LabeledLDA [RCE+11], to make use of a large list of entities gathered from Freebase [BEP+08], an open-domain ontology, with large amounts of unlabeled data in learning.

2.6 NLP toolkits

2.6.1 Stanford CoreNLP

Widely used as reference in NLP tasks, the Stanford CoreNLP[1] provides a set of natural language processing tools. As stated in [Cor16] “it can give the base forms of words, their parts-of-speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, mark up the structure of sentences in terms of phrases and word dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract particular or open-class relations between entity mentions, get quotes people said, etc".

The Stanford NER[2] is able to detect the following named entities: PERSON, LOCATION, ORGANIZATION, MISC, MONEY, NUMBER, ORDINAL, PERCENT, DATE, TIME, DURATION, SET [Cor16].

Stanford CoreNLP can be used from the command-line, from its Java API or using third party APIs when using other programming languages [Cor16] (Python, Ruby, Perl, Scala, Clojure, Javascript (node.js), and .NET languages [MSB+14]) . Compared to other frameworks, it is said to be simple to set up and use [PGOOA16].

[1]http://stanfordnlp.github.io/CoreNLP/

[2]http://nlp.stanford.edu/software/CRF-NER.shtml

At the tokenization level, CoreNLP tries to mimic the Penn Treebank 3 tokenization [MMS93] and implements a Maximum Entropy Model. At the NER level, CoreNLP uses a Conditional Random Field (CRF) and is trained with the CoNLL-2003 dataset.

2.6.2 NLTK

NLTK[3] is another toolkit, written in Python, for Natural Language processing, gathering not only program modules (tokenizers, taggers, chunkers, entity recognizers) but also different datasets, lexicons and tutorials.

One of the primary principles of NLTK, besides simplicity, consistency and extensibility, was modularity. Modularity provides components that can be used independently [BKL09]. This might be specially useful to tune only specific parts of the pipeline or even use third parties in conjunction with this toolkit. Since NLTK uses Python, other Python libraries can also be used to complement it, such as Scikits.learn, which includes algorithms for machine learning and preprocessing.

The efficiency of NLTK in terms of runtime performance is not highly optimized, since the goal of NLTK was to provide a simple and easy to understand out of the box tool, mainly for learning purposes [BKL09].

2.6.3 OpenNLP

The Apache OpenNLP[4] [Fou] is another natural language processing library written in Java, but it can also be used in R using the openNLP package.

It is machine learning based and supports tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and co-reference resolution.

With this toolkit it is possible to either use out-of-the-box pre-trained models or to train a Perceptron or a Maximum Entropy Model, for each of the mentioned NLP tasks. POS tagging and chunking make use of the Penn Treebank tagset, and the Chunker is trained on the CoNLL-2000 dataset [PGOOA16].

OpenNLP is able to detect entities of the types PERSON, LOCATION, ORGANIZATION, TIME, DATE and PERCENTAGE, when using the pre-trained models provided.

2.6.4 GATE - ANNIE

GATE[5] is an open source software for solving text processing problems, including Information Extraction, which is done by a built-in IE component set called ANNIE [CMBT02].

This component includes a Sentence Splitter (which tries to identify individual sentences in documents), a POS Tagger and three different Named Entity Recognition resources:

- ANNIE Gazetteer is a dictionary-based NER engine.

[3] http://www.nltk.org/book/
[4] https://opennlp.apache.org/
[5] https://gate.ac.uk

- ANNIE NE Transducer is rule-based NER
- ANNIE OrthoMatcher is rule-based NER

With this tool it is possible to add new entity types manually, besides the traditional entities PERSON, LOCATION, ORGANIZATION.

ANNIE components can be used independently or even in conjunction with customized modules in order to create new applications [BDF+13].

2.6.5 Twitter NLP tools

Ritter et al. [RCE+11] rebuilt the NLP pipeline (POS tagging, chunking and NER), for the specific purpose of dealing with Tweets.

The novelty of this pipeline is that it “leverages the redundancy inherent in tweets" outperforming this way the Stanford NER system, by training tools on unlabeled in-domain and out-of-domain data [RCE+11].

This NER tool can deal with distinctive named entity types, by using Freebase and LabeledLDA. Some of these distinctive types are PRODUCT, TV SHOW and FACILITY.

2.6.6 TwitIE

TwitIE is a full GATE pipeline, including [BDF+13] social media data Language identification, Twitter tokenizer, Twitter part-of-speech tagger and Text normalization.

It is another open-source NLP tool customized for Twitter text at every module of the pipeline [BDF+13], based on GATE’s ANNIE algorithms.

As explained before in 2.6.4, ANNIE’s modules can be integrated with new customized ones. That being said, TwitIE makes use of the default ANNIE’s sentence splitter (given that most tweets consist in only one sentence, changing this module would not lead to significant improvements) and name gazetteer (which includes lists of cities, organizations, days of the week, etc.), but the other components of the pipeline have been adapted.

The first step of the pipeline is language identification. Then for the tokenization module, TwitIE follows Ritter’s approach [RCE+11][BDF+13]. The following two steps are sentence splitting and gazetteer look-up, which are the unmodified ANNIE’s modules. After that, there is the normalizer, which is basically a spell-corrector designed for social media texts. Then comes the POS tagger, an adapted version of the Stanford POS tagger. All these modifications proved useful for the final component of the pipeline, NER, which has performed better than Ritter’s approach [BDF+13].

2.6.7 Performance comparison

Table 2.1, taken from a study of different NER toolkits by Pinto et al. [PGOOA16], presents the precision, recall and F1 scores of five different toolkits (NLTK, OpenNLP, CoreNLP, TwitterNLP, TwitIE) when applied on two different datasets, the one used in CoNLL-2003 shared task (Reuters corpus) and the one used by Ritter et al. [RCE$^+$11].

Table 2.1: NER toolkit performance comparison for 2 different datasets [PGOOA16]

Task	NER					
Data set	CoNLL			Alan Ritter - Twitter		
Tool	$P \pm \sigma$	$R \pm \sigma$	$F1 \pm \sigma$	$P \pm \sigma$	$R \pm \sigma$	$F1 \pm \sigma$
NTLK	0.88 ± 0.12	0.89 ± 0.11	0.89 ± 0.11	0.77 ± 0.16	0.84 ± 0.13	0.80 ± 0.15
OpenNLP	0.88 ± 0.09	0.88 ± 0.08	0.88 ± 0.08	0.85 ± 0.14	0.90 ± 0.11	0.87 ± 0.12
CoreNLP	0.70 ± 0.30	0.70 ± 0.30	0.70 ± 0.30	0.87 ± 0.15	0.89 ± 0.14	0.88 ± 0.15
Pattern	n/a	n/a	n/a	n/a	n/a	n/a
TweetNLP	n/a	n/a	n/a	n/a	n/a	n/a
TwitterNLP	0.88 ± 0.11	0.89 ± 0.10	0.88 ± 0.11	0.96 ± 0.07	0.97 ± 0.05	0.97 ± 0.06
TwitIE	0.74 ± 0.16	0.80 ± 0.14	0.77 ± 0.15	0.77 ± 0.17	0.83 ± 0.14	0.80 ± 0.15

Most of these tools implement different algorithms to perform NER, and their performances on different entity types varies significantly [AL13, RBBL12, PGOOA16]. Moreover, it is common to find disagreements between these tools regarding specific tokens and their corresponding named entity.

2.6.8 Ensemble methods in NER

While ensemble methods have been proposed in literature for the task of NER, usually these methods were applied at the level of the machine learning algorithms, rather than at the level of ready-to-use toolkits. An example of previous use of ensemble methods for NER, proposed by Wu, Chia-Wei, et al. [WJTH06], consisted in applying a memory-based ensemble method on Chinese datasets to achieve better results than using individual classifiers. Another example of the same use was proposed by S. Saha and A. Ekbal [SE13], once again showing that combining different learning algorithms can improve the performance of Named Entity Recognition.

To the best of our knowledge, there have been few attempts to simultaneously use different out-of-the-box toolkits to perform Named Entity Recognition on social media texts. The idea of combining toolkits was applied in one of the submissions to the Making Sense of Microposts challenge in 2013 [CBVR$^+$13]. In this study the authors combined different toolkits using machine learning techniques, and their results showed that several classification models could achieve better results than the best individual tools [CBVR$^+$13].

A more recent example of combining toolkits used two different toolkits (SpaCy and CoreNLP) together to create an hybrid NER tool [JBL16].

2.7 Conclusions

Named entity recognition has been widely studied in recent years. Multiple widely known NLP toolkits are available for free use, such as the Stanford CoreNLP, OpenNLP and NLTK.

While most of these tools have shown solid performance on different, generally formal, text corporas, there has also been a consistent decrease in performance when applying these tools to texts extracted from social media.

Several measures have been proposed, studied and shown to improve performances in different parts of the pipeline, most of them by using techniques to normalize the irregularities of this type of text. However, information about these techniques is sparse and few studies have tried to summarize them and do explicit comparisons between their performances. Many of these techniques have also been tried independently from each other, and the possible co-existence of different techniques has not been studied comprehensively.

A few general NLP tools have been assembled for social media though, examples of that being the Twitter NLP tools by Ritter et al. [RCE+11], an entire pipeline developed specifically for twitter posts, and the TwitIE ANNIE's modified pipeline.

It is fair to say that named entity recognition is far from a closed topic, specially in the social media context, given there is still room for significant improvement. However, there is a wide variety of tools available for free use that perform reasonably well, and while some tools have been proved to be robust in many situations and datasets, no tool has been shown to outperform all others in every context.

For the purpose of our work, the main take home messages from this literature review are as follows:

- NER has been widely studied in formal text corpora, and multiple out-of-the-box tools are freely available for use
- NER on social media texts presents great challenges, due to the text informality, which reduce the performance of such tools
- There have been multiple attempts to attenuate these problems, and new out-of-the-box tools have been developed for the specific purpose of NER in social media
- Each tool has specific strengths and weaknesses in terms of entity types
- Ensemble methods have not been extensively studied for out-of-the-box-tools, and to the best of our knowledge, no attempts have been made to combine 4 of the most well-known NER tools (Stanford NER, OpenNLP, Twitter NLP tools, TwitIE) for the task of NER in social media. Our research intends to fill this gap.

Chapter 3

Features in Machine Learning: Literature review

In a classification problem the goal is to obtain a model able to automatically label unseen objects, based on their features or attributes.

The most common approach to induce such model consists in using a machine learning algorithm to learn from some training data, which is basically a set of pre-labeled objects [Seb02]. By looking at previously labeled data the model will hopefully learn how the characteristics (represented by the features) of the objects relate with their label, and from there be able to predict new unlabeled objects.

3.1 Feature Selection

The process of feature selection in classification is important for two main reasons. Firstly, most algorithms perform worse with irrelevant features, and secondly feature selection provides faster training and testing phases.

In some cases, variable dependencies can be important for the predictive power of a classifier. Some of those situations, mentioned in [GE03] are:

- “Noise reduction and consequently better class separation may be obtained by adding variables that are presumably redundant. Variables that are independently and identically distributed are not truly redundant."
- “a variable that is completely useless by itself can provide a significant performance improvement when taken with others."
- “Two variables that are useless by themselves can be useful together."

For the reasons mentioned above, the process of finding subsets of features that together have good predictive power [GE03] can be useful.

3.1.1 Filter Methods

Filter methods remove features based on their characteristics and independently of the predictive algorithm to be used. They are usually faster than other approaches like wrappers and can also be used as a preprocessing step to reduce space dimensionality and avoid overfitting [GE03].

Filter methods can be either univariate or multivariate. The former ignores feature dependencies and redundancy since only one feature is analyzed at a time. On text classification problems, this drawback assumes even greater importance, given that text classifiers are usually of multivariate nature, i.e, usually their analysis is based on a combination of features [For07].

On the other hand, this also means that univariate filters are even faster than multivariate filters. Two examples of filter methods, explained by Sánchez-Maroño et al. [SMABTS07] are:

- **RelieF**

  This algorithm is an example of a multivariate filter [GE03], which estimates the quality of features based on how well they differentiate two objects from different classes that are near each other. This is done by randomly selecting an object, and then calculating the Euclidean distance to find its nearest same-class object, called "nearest hit", and the nearest different-class object, called “nearest miss". The weights for each feature are then updated according to the difference between the object, the nearest hit and the nearest miss [SMABTS07] [LM07].

- **Correlation-based**

  This algorithm ranks a feature subset based both on the correlation of its features with the class and the correlation between its features. Essentially, a feature will be more relevant if it has a high correlation with the class but not with other features, ensuring this way that each feature "predicts classes in areas of the instance space not already predicted by other features" [SMABTS07].

The advantages of Filter methods [KQ13] are that they work well even on high-dimensional datasets, they are fast and computationally simple, and they are independent of learning algorithms (can also be seen as disadvantage), which is also good since this way feature selection only happens once, and then different classifiers can be evaluated.

3.1.2 Wrapper Methods

Differently from filter methods, wrapper methods are tested for a given predictive algorithm. The algorithm is run with multiple subsets of features and the one that achieves best performance (within that algorithm) is chosen. Therefore, the best feature subset depends on the algorithm to be used.

- **Exhaustive Search (forward and backward)**

  As the name suggests, this method consists in searching all possible feature subsets. The problem with this approach is that it suffers from combinatorial explosion, since a dataset with n features has a total of $2^n - 1$ possible subsets. For a high number of features this approach becomes very expensive computationally. For a small number of features, intuitively it would seem better to perform an exhaustive search, instead of other heuristics, since an optimal solution would be guaranteed that way. However, research shows that exhaustive search can also lead to data overfitting, and thus heuristic searches might be preferred in some cases [For07].

  Exhaustive search could be done by forward selection (begins with empty subset and adds relevant features to the subset iteratively) or backward selection (begins with a full set of features and removes features iteratively).

- **Sequential Forward Selection**

  Sequential Forward Selection (SFS) is an example of an heuristic method. As in the exhaustive forward selection approach, this method begins with an empty subset and iteratively adds features that are relevant to the algorithm.

  The major difference between SFS and the exhaustive forward selection method is that in this case the algorithm stops when some termination criteria is met.

  Examples are stopping the algorithm when adding a new feature does not improve the performance of the algorithm [LM07] or when the increase is smaller than a given threshold.

  One problem with this approach is that once a feature is added to the subset, it cannot be removed. Another problem is that, being an heuristic method, this method may get "stuck" in a local optima.

- **Sequential Backward Selection**

  Sequential Backward Selection (SBS) is another example of an heuristic method. Differently from SFS, SBS starts with all the features present in the subset and iteratively eliminates features [GE03]. It stops when removing a feature does not improve the performance of the algorithm. Similarly to SFS, in SBS once a feature is removed from the subset that same feature will not be added back.

  The drawbacks of these approach are the same as the drawbacks of SFS, namely the risk of getting stuck on local optima.

The methods above are examples of deterministic wrapper methods. There are other non-deterministic approaches, such as genetic algorithms and simulated annealing, which are usually less prone to local optima. The negative side is that they are also more computationally expensive than deterministic methods. [LD11]

In general, wrapper methods have a higher risk of overfitting than filtering methods, and are more computationally expensive [LD11], but they perform better than filter methods.

3.1.3 Embedded Methods

While wrappers use machine learning in a black box manner, embedded methods incorporate variable selection as part of the training process [GE03].

Embedded methods try to optimize two things at the same time, the goodness-of-fit term and a penalty for a large number of variables. These methods are usually specific to a given learning algorithm [GE03].

According to [GE03], embedded methods are more efficient than wrappers for two reasons. First, the data does not need to be split into training and validation sets, leading to more data available. Second, they are also faster in finding a solution since they avoid retraining classifiers for each possible subset.

3.1.4 Summary

Table 3.1 table provides a side-by-side summarized comparison of 3 different feature selection methods.

Table 3.1: Feature selection methods

	Filter methods	**Wrapper methods**	**Embedded methods**
Criteria	Feature (subset) relevance	Feature subset usefulness	Feature subset usefulness
Search	Order features (ranking or nested subsets of features)	Search the feature space	Search guided by the learning process
Assessment	Statistical tests	Cross-validation	Cross-validation
Advantages	Robust against overfitting	Likely to find the most "useful" features	Likely to find the most "useful" features, less expensive, less prone to overfitting
Disadvantages	May fail to select the most useful features	Prone to overfitting	-

3.2 Feature Construction

Feature construction aims at finding better ways of representing the original data (for example to compact data), or generating features more efficient for making predictions [GE03].

It can be done not only by manually adding some specific domain knowledge to the input, but also by using generic algorithms for feature construction [GE03].

An example of a feature construction method is clustering. In this method the idea is to group similar features, generating a cluster, and then using its centroid as a new feature. This is frequently used for bag of words representations of labeled texts, where each word corresponds to a feature. These features are then grouped in a cluster, and the new features become categories of words, instead of words [GE03].

Another example of feature construction happens in the hidden units of artificial neural networks. Each layer of the network constructs more abstracted features from the input features.

While feature construction is often associated with the idea of dimensionality reduction, it can also be used to expand the feature space, by for exampling computing products of the original features to create monomials (Non-linear expansions) [GE06]. This can be useful for very complex problems where the initial features are not enough for the model to learn properly.

3.3 Conclusions

One of the most fundamental parts of a classification problem relies on having the right features.

Feature selection methods serve essentially three purposes: reduce the feature set, in order to save resources; improve the performance; make data more understandable. There is no feature selection method better than all others, and the choice on which to use will depend on the specific problem addressed.

Feature construction may be seen as a method to reduce dimensionality while keeping important information, or even as a preprocessing step. However, they may also be used to expand the feature set in cases where the data is too complex to learn with the initial features.

In our study we will use two wrapper methods, Sequential Forward Selection and Sequential Backward Selection. The reason for this choice was based on the availability of such methods among the software and tools used.

Chapter 4

Named Entity Recognition on Social Media Texts

4.1 Problem definition

The fundamental goal of this thesis depends on the ability to perform named entity recognition on a dataset of social media posts.

As explained in detail in 2.5.2 , Named Entity Recognition on social media posts presents great challenges mainly due to the inherent informality of the content of such posts.

The first part of this thesis consists therefore in finding a way to address these challenges, and building or finding a Named Entity Recognizer able to extract entities from the dataset in the most efficient way.

Therefore, the following research questions arise:

- Which is the best available tool for Named Entity Recognition in social media?
- How can we improve such tool(s)?

4.2 Solution outline

Our solution to this problem begins therefore with the exploration of different out-of-the-box NER tools.

As concluded in 2, multiple text normalization techniques have been proposed that can be used to improve the accuracy of named entity recognition in informal texts, many of them included in entire NLP pipelines designed for this specific task, such as the Twitter NLP tools [RCE+11] (with part-of-speech tagging, chunking and named-entity recognition) and Gate with TwitIE plugin [BDF+13] (a sequence of modules including language identification, tokenization, spelling and orthographic corrector, Stanford POS tagger adapted to Twitter, and a Named Entity Recognizer).

Besides these two twitter-specific NLP pipelines, we decided to still use two traditional systems: CoreNLP and OpenNLP.

The reason to do so is because the results presented in different studies may differ in the presence of a completely new dataset, as will be the case in this work. For example, according to a toolkit performance comparison by Pinto et al.[PGOOA16], both CoreNLP and OpenNLP actually performed better than the Ritter et al.'s [RCE+11] approach - which was proposed specifically for twitter posts - on a twitter dataset. These examples of unexpected results support the idea that no tool has a clear superiority over others in every context, and therefore analyzing as many tools as possible can be important. This step includes:

- Finding appropriate annotated datasets of social media posts
- Setting up and experimenting Stanford NER
- Setting up and experimenting GATE (with the TwitIE plugin)
- Setting up and experimenting OpenNLP
- Setting up and experimenting Ritter et al. approach [RCE+11]

We could settle for the best-performing tool available and use that one for our further experiments. However, as explained in section 2.6.7 these tools have different behaviours regarding different entity types, and it is common to find disagreements between these tools. Therefore, it was our intuition that the simultaneous use of different toolkits might help achieve better results than using them separately. Apart from the obvious benefit that some of these toolkits predict different sets of entity types, complementing each other that way, we will analyze, for a standard set of core entities (PERSON, LOCATION, ORGANIZATION), if a ponderation between toolkits reveals to be beneficial.

That being said, in this chapter we study the combined use of four different NLP toolkits — Stanford CoreNLP, GATE, OpenNLP and Twitter NLP tools — in the context of social media. Previous studies have shown performance comparisons between these tools, both on news and social media corpora. Here, we go further by trying to understand how differently these toolkits predict Named Entities, in terms of their precision and recall for three different entity types, and how they can complement each other in this task in order to achieve a combined performance superior to each individual one.

The proposed solution to the problem, which from now on will be referred to as the "Ensemble of NER tools" brought two new research questions:

- Can an ensemble of different toolkits achieve overall higher NER performance than any of the involved toolkits, independently, for the same task?
- What is the best way to resolve conflicts/disagreements between different toolkits regarding their entity predictions?

The remainder of this chapter is presented as follows: in section 4.3 we describe our experimental setup, including details on the datasets used, brief descriptions of the toolkits used and the necessary steps taken to obtain their results for our analysis, the ensemble itself and the different voting protocols tested, as well as the performance measures used; in 4.4 we present the results and discuss them in detail; finally, in section 4.7 we present our conclusions and ideas for future work.

4.3 Experimental Setup

4.3.1 Datasets

For comparison purposes, every toolkit used equally pre-tokenized datasets, following Ritter's [RCE+11] tokenization method. We also chose to focus only on the entities PERSON, LOCATION and ORGANIZATION. The reason for this choice was that these entities are the only three entities detected by all the toolkits tested.

For the first experiment, an original dataset of tweets from our project [FSF16] was partially used. This dataset consists of 840 entries: 420 tweets, 107 Facebook posts and 313 Facebook comments, retrieved by a crawler about 6 topics highly discussed in 2016: “Refugees Syria", “Elections US", “Olympic Games" , “Terrorism" , “Daesh" and “Referendum UK EU".

This original dataset was then tokenized. Therefore, instead of 840 entries, the tokenized dataset had 28172 entries (one per token). From the tokenized dataset, a subset of 3474 tokens was extracted. The final dataset contains one token per row, and one entity for each token. The ground truth for this dataset was manually annotated by the authors.

For the second experiment, a dataset from “WNUT NER - Workshop on Noisy User-generated Text" [BDMH+15] - was used. This dataset used the same format seen in Twitter NLP tools by Ritter et al., including less common entity types that were dropped for the purpose of this study, which focuses only on the 3 core entities PERSON, LOCATION and ORGANIZATION.

In the third experiment, we tested the dataset from the 3rd workshop on “Making Sense of Microposts" (#MSM13) [CBVR+13], which took place in 2013. It is important to note that for this dataset we used the PTBTokenizer available as part of the CoreNLP libraries. The reason for this choice was that in the conversion process we had to tokenize both the entities and the text of the tweets, and for the tokenizations to match we needed a deterministic tokenizer.

Therefore, our testing datasets are (each entry corresponds to a token):

- **Dataset 1**: Project REMINDS - 3474 entries
- **Dataset 2**: WNUT NER - 48 862 entries
- **Dataset 3**: #MSM2013 - 62 494 entries
- **Dataset 4**: Subset of #MSM2013 - 10 000 entries

Every dataset used in this study (excluding Dataset 2, which we are not allowed to distribute) is available in a public repository created for the purpose: `https://bitbucket.org/filipebatistaner/ensemblener`

4.3.2 Toolkits and Data preparation

4.3.2.1 Stanford CoreNLP[1]

Stanford CoreNLP was run using the default toolkit via command line [MSB+14]. This toolkit accepts as input format the tokenized text and the output format in a tab formatted file, convenient for this study.

Since there was not enough labeled data for training our own model, as the data was manually annotated by the authors and that is a very costly task timewise, we used the "3 class model" provided by CoreNLP, which was trained on both MUC 6 and MUC 7 training data sets with some additional data (including ACE 2002 and other generated data).

4.3.2.2 GATE using TwitIE plugin[2]

GATE provides a graphical interface which was used in this study to run the TwitIE [BDF+13] pipeline, available as part of the Twitter plugin.

The output format consists in surrounding any detected entity with XML tags. In order to convert this type of output to the tab separated format, a small script using regular expressions was written in Python.

While GATE is able to detect many other entity types, we used only the three core entities (PERSON, LOCATION and ORGANIZATION). We used the default configurations of the TwitIE pipeline.

4.3.2.3 Twitter NLP tools[3]

Twitter NLP tools was run via command line, following the usage presented in the Twitter NLP tools Github repository.

Twitter NLP tools' [RCE+11] output is by default in the IOB format [RM99] (B for beginning of a Named Entity (NE), I for inside an NE, O for outside of NE), and the "token/ENTITY" format. The IOB format was dropped, so instead of B-ENTITY and I-ENTITY we opted to use ENTITY only. Besides, 2 entity types were converted: COMPANY to ORGANIZATION, and GEO-LOCATION to LOCATION, while all the remaining entity types (except PERSON) were simply dropped.

We used this tool as is, without any re-training or tuning.

[1]https://stanfordnlp.github.io/CoreNLP/
[2]https://gate.ac.uk/download/
[3]`https://github.com/aritter/twitter_nlp`

4.3.2.4 OpenNLP[4]

OpenNLP is a Java library which supports several common NLP tasks, including Named Entity Recognition.

OpenNLP can be used directly as a tool, or via its API. We decided to use the API in a small Java project in order to easily output the entities to the tab-separated format.

We used the pre-trained models for the OpenNLP 1.5 series, for each entity type used.

4.3.3 Ensemble voting methods

In order to study the viability of a NER toolkit ensemble, all the outputs from the previous toolkits previously mentioned were merged to a single *comma-separated values* file, one column for the tokens, another column for the ground truth entities, and one column for each of the entities predicted from each toolkit.

The first step was to compute the precision, recall and F1 measure for each toolkit individually, using the ground truth obtained by manual labeling.

The second step was to define different voting protocols to resolve the conflicts between the different toolkits predictions.

Finally, we used different machine learning algorithms taking as input features the predictions of each tool.

4.3.3.1 Protocol use 1:

A token is tagged with entity type *A* if and only if at least one of the following conditions are met:

- 50% of the toolkits predicted entity type *A* and the other 50% did not predict any entity type
- At least 75% of the toolkits predicted entity type *A*

4.3.3.2 Protocol use 2:

A token is tagged with entity type *A* if and only if at least one of the following conditions are met:

- 50% of the toolkits predicted entity type *A* and the other 50% did not agree on any other entity between them.
- At least 75% of the toolkits predicted entity type *A*

4.3.3.3 Machine learning approach:

The models for predicting the combined output were obtained by running each of the following ML algorithms on a training set, with 10-fold cross validation, and then tested on an independent test set.

[4]https://opennlp.apache.org/

Both the train and test sets were subsets, each of 10 000 entries, of the previously mentioned MSM2013 dataset. Every ML experiment was performed in RapidMiner Studio. The algorithms used were Naïve Bayes, Random Forest, k-nearest neighbors (k-NN) and Neural Network. The features used consisted of the 4 individual outputs of each tool.

4.4 Experimental results

In this section we explore the performances of each toolkit and compare them to the Ensembles' performances using different protocols and datasets. $Ensemble_n$ (E_n) will be the notation used to refer to the Ensemble using protocol number n, previously defined. Bold will be used to highlight the highest result, and "Avg." corresponds to the average of the three entity types.

For each dataset we provide an extensive analysis, providing not only the F1 score results but also the Precision and Recall (these metrics were explained in detail in 2.4). In this study we chose to use the lenient way of counting true positives (i.e. we consider that partially correct or overlap annotations are correct).

For dataset 4 more experiments were added using ML algorithms. This type of experiments was not conducted on the other datasets.

4.4.1 Dataset 1 - REMINDS

4.4.1.1 F1 score analysis

Table 4.1: F_1 scores on Dataset 1

	PERSON	LOCATION	ORGANIZATION	Avg.
CoreNLP	67.37	78.26	28.57	58.07
TwitIE	67.5	85.87	**44.90**	**66.09**
TwitterNLP	44.90	72.84	10.39	42.71
OpenNLP	63.63	67.95	23.08	51.55
$Ensemble_1$	**80.00**	84.52	30.59	65.04
$Ensemble_2$	74.07	**88.14**	30.59	64.27

Ensemble 1 Looking at Table 4.1 and Figure 4.1 we can see that $Ensemble_1$ achieved the highest F-measure for detecting the entity PERSON.

In terms of LOCATION and ORGANIZATION entities, while $Ensemble_1$ was better than CoreNLP, Twitter NLP tools and OpenNLP, it did not perform better than TwitIE.

On average, TwitIE still achieved the best F1 measure, with 66%, followed immediately by the ensemble, which achieved an average F_1 of 65%. This is not surprising, given that TwitIE was, among all the 4 toolkits, the one to achieve better results for every entity type.

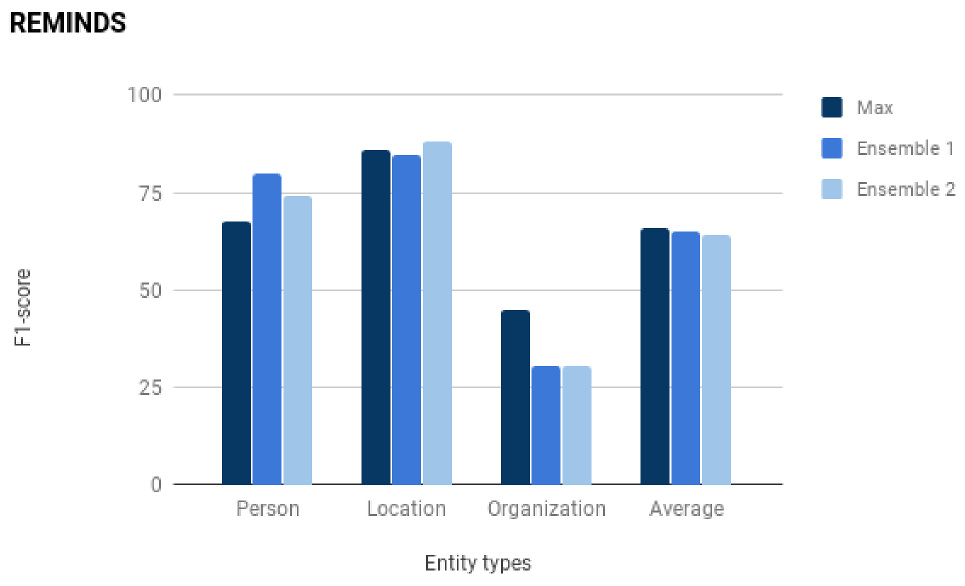


Figure 4.1: F1-scores of the entities Person, Location and Organization on the REMINDS dataset

Nevertheless, it was possible to achieve an improvement of 12.5% on the detection of the entity Person by using $Ensemble_1$.

Ensemble 2 $Ensemble_2$ also achieved the best F_1 score for the entity type PERSON when compared to any other toolkit individually, however its F_1 score was lower than $Ensemble_1$.

On the other hand, $Ensemble_2$ scored higher than $Ensemble_1$ and any other toolkit and in terms of detecting the entity LOCATION.

On average, $Ensemble_2$ was worse than $Ensemble_1$, which in turn was worse than TwitIE.

While for this dataset our ensembles did not outperform the best individual toolkit, TwitIE, there were still visible improvements in specific entity types, namely PERSON and LOCATION.

4.4.1.2 Recall analysis

Ensemble 1 In terms of recall, it is possible to see in Table 4.2 that the $Ensemble_1$ ranked second for every entity type. The toolkit with highest recall for the entity PERSON was the Stanford CoreNLP, while TwitIE was the toolkit to achieve highest recall for the entities LOCATION and ORGANIZATION.

Ensemble 2 $Ensemble_2$ ranked better than $Ensemble_1$ for the entity type LOCATION, but scored the same for PERSON and ORGANIZATION.

The fact that protocol 2 was less strict than protocol 1 is the likely reason for the improve in recall from $Ensemble_1$ to $Ensemble_2$

Table 4.2: Recall scores on Dataset 1

	PERSON	LOCATION	ORGANIZATION	Avg.
CoreNLP	**80**	65.63	16.67	54.1
TwitIE	67.5	**82.29**	**30.56**	**60.12**
TwitterNLP	55	61.46	5.56	40.67
OpenNLP	52.5	55.21	16.67	41.46
$Ensemble_1$	75	73.96	18.06	55.67
$Ensemble_2$	75	81.25	18.06	58.10

4.4.1.3 Precision analysis

Table 4.3: Precision scores on Dataset 1

	PERSON	LOCATION	ORGANIZATION	Avg.
CoreNLP	58.18	96.92	**100**	85.03
TwitIE	67.5	89.77	84.62	80.63
TwitterNLP	37.93	88.33	80	69.11
OpenNLP	80.77	88.33	37.5	63.13
$Ensemble_1$	**85.71**	**98.61**	**100**	**94.79**
$Ensemble_2$	73.17	96.30	**100**	89.82

Ensemble 1 In terms of precision, $Ensemble_1$ ranked first for the three entity types, as we can see in Table 4.3. This result makes sense and indicates that using this protocol helped significantly in detecting entities efficiently, by eliminating predictions with less than a certain level of confidence (see protocol 1).

Ensemble 2 $Ensemble_2$ overall precision dropped when compared to $Ensemble_1$, 12.54% on PERSON and 2.31% on ORGANIZATION. Once again it makes sense that reducing the strictness of the protocol would likely reduce the precision.

4.4.2 Dataset 2 - WNUT NER

In Table 4.4 and chart 4.2 we can see that for this dataset the results were generally low for all the toolkits, when compared to the performances obtained from the other datasets tested. Since this dataset used Twitter NLP tools format, it had to suffer the same conversion explained in Section 4.3.2.3, which probably led to the worse results.

Table 4.4: F1-scores on Dataset 2

	PERSON	LOCATION	ORGANIZATION	Avg.
CoreNLP	56.62	32.5	20	36.37
TwitIE	59.95	**48.14**	38.23	48.77
TwitterNLP	52.78	34.9	**45.12**	44.27
OpenNLP	43	34.79	6.59	28.13
Ensemble$_1$	**70.57**	41.45	42.37	51.46
Ensemble$_2$	70.44	44.53	41.73	**52.53**

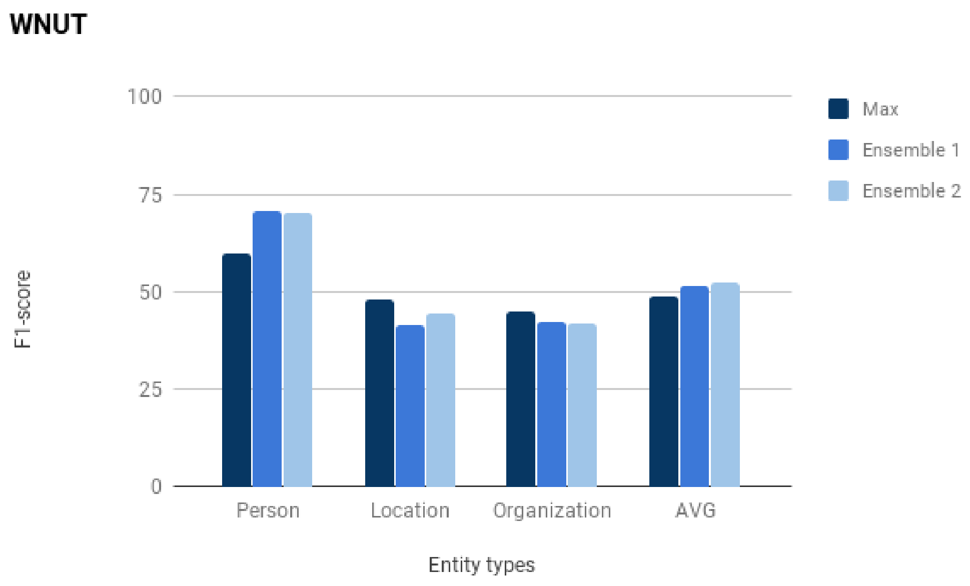


Figure 4.2: F1-scores of the entities Person, Location and Organization on the WNUT dataset

Nevertheless, we can see that both Ensembles achieved better F-scores on average than any other toolkit alone, which is the question we sought to answer in this work.

In terms of precision (Table A.2) and recall (Table A.2 in Appendix A) , the conclusions were the same as for Dataset 1: the stricter protocol ($Ensemble_1$) had less recall but more precision than the less strict protocol ($Ensemble_2$).

4.4.3 Dataset 3 - #MSM2013

Table 4.5: F1 scores on Dataset 3

	PERSON	LOCATION	ORGANIZATION	Avg.
CoreNLP	69.20	54.18	27.09	50.16
TwitIE	77.06	**67.96**	43.95	62.99
TwitterNLP	55.04	41.91	16.18	37.71
OpenNLP	55.40	47.68	25.47	42.85
$Ensemble_1$	79.93	62.20	41.37	61.17
$Ensemble_2$	**82.36**	66.42	**45.26**	**64.68**

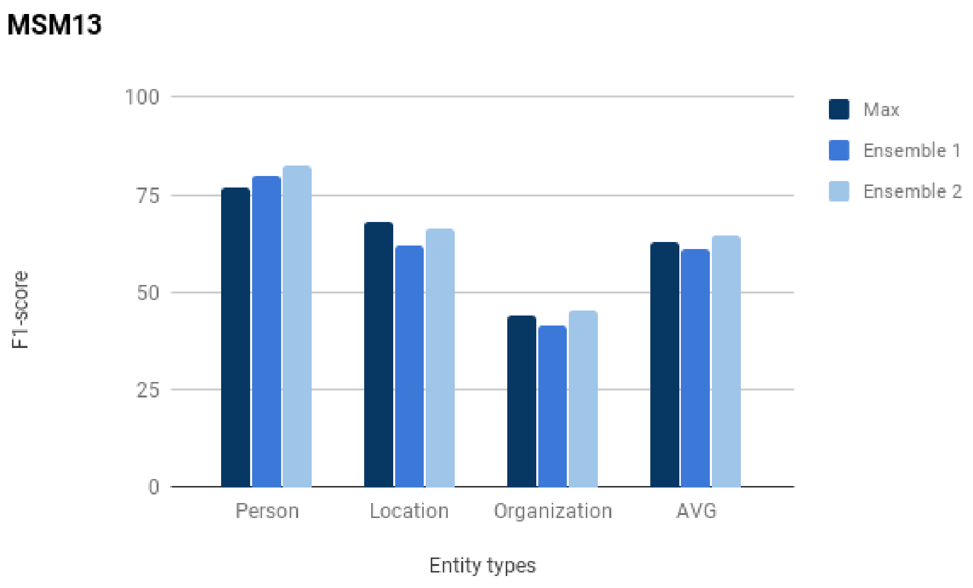


Figure 4.3: F1-scores of the entities Person, Location and Organization on the MSM13 dataset

Looking at Table 4.5 and Figure 4.3 it is possible to see that once again ensemble 2 performed better on average than any other toolkit individually. Ensemble 2, while not better than TwitIE on average still performed reasonably well with only 1.82% less F1-score.

Also, once again, both Ensembles outperformed every toolkit on the entity type PERSON, and Ensemble 2 on the entity type ORGANIZATION.

Once again, the conclusions in terms of Precision and Recall remained the same. As we can see in Tables A.3 and A.4 in Appendix A, $Ensemble_1$ achieved higher precision and less recall than $Ensemble_2$.

4.4.4 Dataset 4 - Subset of #MSM2013

Table 4.6: F1 scores on Dataset 4

	PERSON	LOCATION	ORGANIZATION	Avg.
CoreNLP	54.21	65.64	40.20	53.35
TwitIE	71.13	82.20	**61.02**	71.45
TwitterNLP	51.73	55.59	15.85	41.06
OpenNLP	52.80	63.05	41.20	52.35
$Ensemble_1$	80.08	77.07	53.57	70.24
$Ensemble_2$	**81.26**	81.45	57.74	73.48
Random Forest	80.68	82.58	51.4	71.55
Naive Bayes	80.88	83.82	57.37	**74.02**
kNN, k=3	75.09	**84.17**	47.76	69.00
kNN, k=10	80.68	82.53	53.16	72.12
Neural net	79.62	83.71	58.68	74.00

We extracted a subset of 20000 tokens from #MSM2013 and split it into two equally sized datasets for training and testing purposes. For a fair comparison, we ran our manually defined protocols and the individual toolkits again on the test set.

Looking at Table 4.6, we can see that Naive Bayes was the best method on average (74.02% F1), followed by the Neural Network (74.00% F1), and our manually defined $Ensemble_2$ (73.48% F1). Every ML algorithm that we experimented, except kNN with k=3, performed better than TwitIE (the best among the tools).

In terms of individual entity types, our $Ensemble_2$ was the best for PERSON, achieving 81.26% of F1, an improvement of 10.13% against TwitIE. For LOCATION, the best achieved was 84.17%, using kNN with k=3, an increase of 1.97% (again against TwitIE). For the entity type ORGANIZATION none of our ensembles was able to perform better than TwitIE.

An interesting fact to note is that the best Ensemble on average (Naive Bayes) was not the best Ensemble for any specific entity type alone.

4.5 Results summary

Differently from results previously shown in literature [RCE+11, PGOOA16], in our experiments Twitter NLP tools achieved overall worse performances than other toolkits across all the 3 tested datasets. We believe this performance difference was related to the way we converted the output of this toolkit for our study. We expose our rationale for this.

Firstly, Twitter NLP recognizes multiple entity types, but those entities do not include ORGANIZATION nor LOCATION. Instead, they include COMPANY and GEO-LOCATION, which were converted directly to ORGANIZATION and LOCATION. We are aware that the former is probably not optimal, since a company does not need to be an organization and vice-versa.

Secondly, there is also the fact that Twitter NLP tools recognizes other entity types that we decided to ignore in this study (such as SPORTSTEAM, BAND, and MOVIE) which could be, in some cases, sub-categories of more general entity types (for example a SPORTSTEAM could be seen as an ORGANIZATION/COMPANY). Therefore, ignoring such entity types could be another reason for the comparatively worse results obtained by Twitter NLP tools in our experiments.

Finally, we did not include optional features based on POS and chunk tags, which leads to faster but lower quality results [RCE+11].

For the first dataset, while TwitIE has remained better than both ensembles on average, we witnessed a positive boost of PERSON detection using Protocol 1, achieving more 12.5% F1-score than the best individual toolkit (TwitIE with 67.5%), and a boost in LOCATION detection using Protocol 2, achieving more 2.27% F1-score than the best individual toolkit (TwitIE with 85.871%).

On the second dataset, both ensembles have beaten the best individual toolkit. The performance boost was very noticeable on the entity type PERSON (up to 10.62%), and the ensembles managed to keep a reasonable performance on the detection of ORGANIZATIONS (42.37% and 41.73% respectively), given that two of the toolkits (CoreNLP and OpenNLP) achieved very low results for this entity type (20% and 6.59% respectively).

In our third experiment, the boost on the entity type person remained noticeable for both ensembles (2.87% and 5.3% higher than the best toolkit). $Ensemble_2$ performed better on average than any other toolkit, achieving 1.69% higher F1-score than TwitIE, the best individual toolkit with 62.99% F1-score.

In terms of precision and recall, the conclusions were the same as for every dataset: the stricter protocol ($Ensemble_1$) had less recall but more precision than the less strict protocol ($Ensemble_2$).

Finally, our last experiment showed that there were some ML algorithms able to outperform TwitIE and even our $Ensemble_2$, namely Naive Bayes and the Neural Network.

4.6 Ensemble tool implementation in Java

All of the above results and experiments were performed manually, by running each toolkit, cleaning and formatting their outputs, merging them in a single file, and measuring their performances using R.

It was our goal, however, to implement an easy-to-use tool with this whole process automatized, in a concurrent multi-threading system.

Since most of these toolkits were implemented in Java, for our Ensemble system we decided to use Java as well. This system receives a tokenized dataset, with or without labels, and outputs both Protocol 1 and Protocol 2 predictions - we used manually defined protocols instead of a Naive Bayes or a Neural network for simplification and efficiency purposes. In case the dataset has labeling, the system also provides Precision, Recall and F1-scores for each toolkit and the protocols implemented.

The developed system has also the capability of generating a dataset with entity features, useful for the next chapter experiments.

4.7 Conclusions

The first conclusion of this study is that using an ensemble of toolkits with a voting system seems to improve the performance of NER on tweets, answering the first question of our research.

As for the second question, we can say that both manually defined protocols were, to some extent, "naive" yet they achieved promising results. This indicates that a more refined protocol will probably improve these results even further. It proves to be false, this approach could still be used with a combination of both protocols for the entities PERSON and LOCATION, and keeping ORGANIZATION predicted by TwitIE. We also showed that using machine learning algorithms for predicting entities based on the outputs of each toolkit is viable.

Additionally, we implemented an automatic "out-of-the-box" Ensemble tool for Named Entity Recognition, with feature generation capabilities, based on the entities extracted.

Chapter 5

Relevance Detection

In this chapter we study the importance of named entities mentioned in a post, namely PERSON, LOCATION and ORGANIZATION for evaluating the relevance of that post.

For this purpose, we extracted entities from a dataset of posts, using our ensemble of NER tools, and then experimented feature extraction and selection methods to analyze if such features could be used effectively (and how effectively) to build a machine learning classifier for detecting relevant and irrelevant posts.

5.1 Problem definition

As explained in the introduction (section 1.2), this thesis is integrated in a relevance detection research project, REMINDS [FSF16]. This project aims to build a system capable of detecting relevant information (in a news-worthy sense of relevance) in social media, namely on tweets and Facebook posts and comments.

This can be seen as a binary classification problem, in which the goal is to classify a post as “relevant" or “not relevant", given a number of different features.

While many different features related to the posts have been previously studied in the context of this project, named entities referred in the text of these posts deserve a more detailed study. Therefore, the contribute of this thesis to this research work is to add new features based on named entities to the current input vector of features, and analyze the impact of such features in the classification system.

Therefore, the following research questions arise:

- **RQ1**: can an automatic system detect such entities? Is it possible to improve the accuracy of existing systems?
- **RQ2**: are named entities useful to predict relevance of social media posts?

- **RQ3**: are statistics related to named entities useful to predict relevance of social media posts?
- **RQ4**: which features are more important for relevance detection?
- **RQ5**: do named entity-based features improve the performance of a classifier when combined with a word-to-vector approach?
- **RQ6**: how does the system behave across different datasets?

The remainder of this chapter is presented as follows: in section 5.2, we describe the datasets used. In section 5.3, we will present a descriptive analysis on the dataset used. In section 5.4 we will explain the different experimental setups and the obtained results. Finally, in section 5.5 we will present our conclusions and answer our research questions.

5.2 The Data

5.2.1 Training Dataset 1

The first dataset used for relevance detection analysis consisted of 35833 tweets, 15680 classified as “relevant" and 20153 as “not relevant".

This classification was done manually, by 5 human annotators. However, instead of classifying each post individually, the classification was based on the authors’ potential of being news-worthy. To decide if a user had potential to post relevant information, i.e, information that could appear in credible newspapers or media, each annotator labeled users by analyzing 10 tweets posted by that user. Then every tweet posted by that user was classified based on the user’s own classification.

The agreement used for the labeling was 4, and the inter-rater agreement was of 86.2%.

5.2.2 Validation dataset

To better ensure the performance of our classifiers, we decided to use a validation dataset in some of our experiments.

We experimented using as validation dataset an external dataset, with tweets obtained from a completely different temporal window, different topics, and using a different labeling system. The validation dataset consisted of 1050 posts from both Facebook and Twitter, manually annotated by 3 human annotators. Each post was annotated by only one person from a set of annotators composed by researchers and University students, who were explained the goal of the task, and therefore there is no agreement or inter-rater agreement.

The use of this validation dataset aimed to analyze the versatility of the system, by testing the possibility of overfitting to the dataset, for example due to common topics and/or users.

5.2.3 Feature extraction

To analyze the importance of Named Entities in relevance detection we started by creating features based on entities extracted from the text of the posts. These entities were extracted using both Stanford NER in some experiments, and using our NER ensemble (previously developed and explained in detail in chapter 4) in other experiments. The entire feature list can be seen in Table 5.1.

5.2.3.1 Internal named entity-based features

These features include statistics of the entities extracted, within the scope of a post , i.e., the number of persons, locations, organizations mentioned in a single post.

Besides these numerical features, we created seven boolean features based on them: the presence of persons, locations, or organizations in a post (3 features), the simultaneous presence of pairs of entity types (3 features), the simultaneous presence of all entity types (1 feature), and the total absence of entities (1 feature).

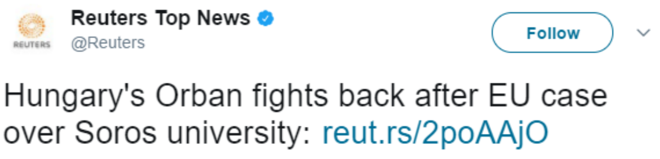
5.2.3.2 External named entity-based features

Since the goal of this investigation consisted in analyzing how named entities help in the task of relevance detection, it seemed only logical that if a specific named entity was common in news articles, its presence in a social media post would potentially indicate the relevance of that post. With that in mind, we decided to weigh named entities based on a “news-worthiness" ranking.

We created this ranking by using the open API provided by *The Guardian*[1]. This API allowed us to retrieve the number of mentions of a given named entity in the news, for a given period of time - we decided to count mentions in the news from 1 month before the date of each post.

Therefore, for each post, six additional features were created, three of them consisting of the sum of the number of mentions in the news for each named entity type, and the remaining three the average number of mentions in the news per each entity type.

For instance:


Reuters Top News
@Reuters
Follow

Hungary's Orban fights back after EU case over Soros university: reut.rs/2poAAjO

Figure 5.1: Example tweet from “Reuters Top News"

Three entities were detected by our ensemble for the tweet above: 1 person entities (Orban), 1 organization entity (EU) and 1 location entity (Hungary).

The tweet was posted in 2017-04-24. Since we counted every mention up to 1 month before the post date, the interval in this case was from 2017-03-27 to 2017-04-27.

[1]http://open-platform.theguardian.com/

There were a total of 11 mentions in the news for the entity “Orban", therefore the n_PER_news is set to 11. Then, there were 32 mentions of the entity “Hungary", therefore the n_LOC_news feature is set to 32. Finally, there were 779 mentions of the entity “EU", therefore n_ORG_news feature is set to 779.

The other three features are simply averages. In this specific example, the rates were the same as the previous features, because there was only one entity per entity type.

5.2.3.3 Word embeddings-based features

Apart from testing entity-based features in isolation, we decided to also try to combine them with word embeddings-based features.

In previous experiments, our relevance detection system obtained maximum performance with these features. Therefore, we wanted to understand if adding entity-based features on top of word embeddings could help improve our system's performance even further.

Word embeddings consist in vector representations of words, i.e., words are mapped to a dense vector of numbers. For our experiments, we used Word2vec, a method proposed by Tomas Mikolov et al. [MCCD13, MSC$^+$13], implemented in the R package “wordVectors"[2]. We trained our own word2vec model using our dataset. We chose to produce 100 vectors, with a window size of 12 words.

Table 5.1: Features list

ID	Type	Description
1	Numerical	Number of persons mentioned in the post
2	Numerical	Number of locations mentioned in the post
3	Numerical	Number of organizations mentioned in the post
4	Numerical	Total number of entities (sum of the previous features) mentioned in the post
5	Boolean	Persons mentioned in the post
6	Boolean	Locations mentioned in the post
7	Boolean	Organizations mentioned in the post
8	Boolean	Persons and locations simultaneously mentioned in the post
9	Boolean	Persons and organizations simultaneously mentioned in the post
10	Boolean	Locations and organizations simultaneously mentioned in the post
11	Boolean	All entity types simultaneously mentioned in the post
12	Boolean	No entities mentioned in the post
13	Numerical	Sum of the number of mentions in the news of all person entities mentioned in the post
14	Numerical	Sum of the number of mentions in the news of all location entities mentioned in the post

[2]https://github.com/bmschmidt/wordVectors

15	Numerical	Sum of the number of mentions in the news of all organization entities mentioned in the post
16	Numerical	Sum of the number of mentions in the news of all entities mentioned in the post
17	Numerical	Average of the number of mentions in the news per person entity mentioned in the post
18	Numerical	Average of the number of mentions in the news per location entity mentioned in the post
19	Numerical	Average of the number of mentions in the news per organization entity mentioned in the post
20	Numerical	Word embeddings (100 features)
L	Boolean	relevance (label)

5.3 Descriptive Analysis

5.3.1 Data distribution

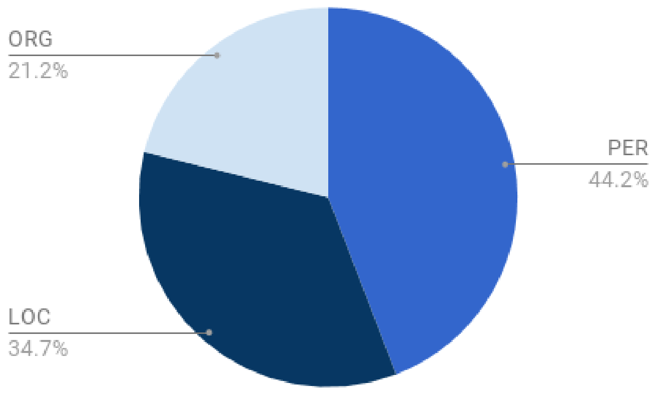


Figure 5.2: Entity type distribution

As we can see in Figure 5.2, the predominant entity type in the dataset is PERSON (with 44.2%), followed by LOCATION (with 34.7%) and ORGANIZATION (with 21.2%).

5.3.2 Feature correlation analysis

In order to analyze the correlation between features, the Pearson Correlation was used.

$$r = \frac{cov(X,Y)}{\sqrt{var(X)}\sqrt{var(Y)}} \tag{5.1}$$

Two correlation matrices were plotted: one for the numerical features (Figure 5.3), and another for the dichotomous (boolean) features (Figure 5.4).

The reason to do such separation was twofold. Firstly, most of the dichotomous features were strongly correlated with the numerical features, which was expected since these features were generated from the numerical features. Secondly, it would not make sense to compare correlation values between pairs of different types: i.e, comparing a numerical-dichotomous correlation with a numerical-numerical correlation, for example. Therefore, there was no point in having all of these variables in the same correlation matrix.

Dichotomous variables were converted from “true" and “false", to 1 and 0 respectively.

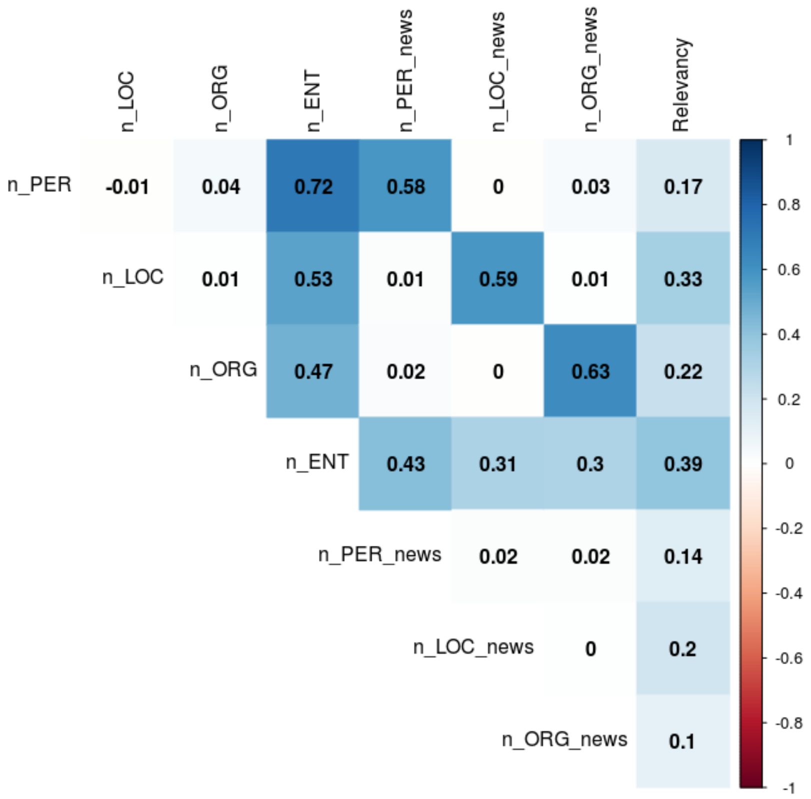


Figure 5.3: Correlation matrix for numerical features

Figure 5.3 presents the correlation coefficients between numerical features. Starting by the correlations for the label (relevance), it is possible to see that the most correlated feature with the label was the total number of entities. This was an expected result, since news-worthy (relevant) posts should intuitively include at least some named entities.

The second most correlated attribute with the relevance was the number of locations mentioned in a post (n_LOC), followed by the number of organizations (n_ORG) and then the number of persons (n_PER).

Neither of these correlation coefficients were strong, however, with all of them standing between 0.17 and 0.33.

For the news-based features, the highest correlation was achieved by location mentions in the news (n_LOC_news), followed by person mentions in the news (n_PER_news) and organization mentions in the news (n_ORG_news). However, these correlations were very weak, scoring a maximum of 0.2.

The remaining correlations show the redundancy between attributes. As expected, the total number of entities showed high correlations with the number of each specific entity type. There were significant correlations between the number of entities and the number of news mentions, a result that was also expected.

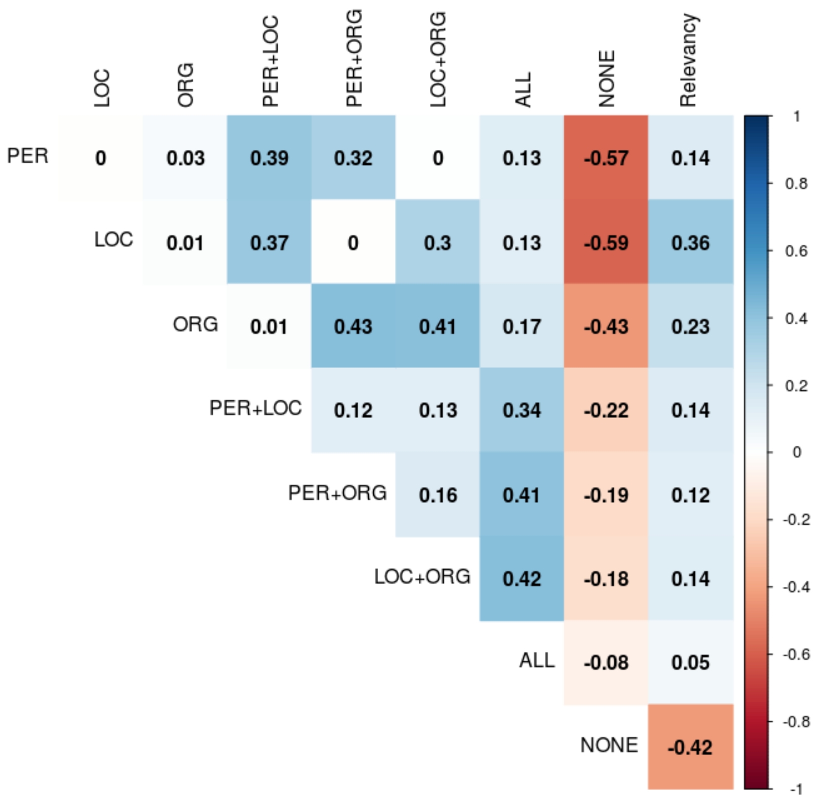


Figure 5.4: Correlation matrix for dichotomous features

Looking at Figure 5.4, new insights about the combination of different entities can be retrieved.

As expected, the absence of entities in posts (represent by the attribute NONE) had a significant negative correlation with their relevance (-0.42). Therefore, it is more correlated with the "not relevant" label.

On the other hand, the simultaneous presence of every entity type (represented by the attribute ALL), had an almost nonexistent correlation with the relevance.

It is also interesting to see that the combination of different entity types had lower correlation with relevance than the independent entity types alone. For example, Person had a correlation coefficient of 0.14, and organization a correlation coefficient of 0.23. When combined (PER+ORG), however, the correlation coefficient was only 0.12. The same applied to the pairs PER_LOC and LOC_ORG.

5.3.3 Feature weights by Information Gain Ratio

Table 5.2: Feature weights by information gain ratio

Feature	Weight
n_locations	0.147
location_mentioned	0.147
n_entities	0.137
none_mentioned	0.137
location_organization_mentioned	0.108
n_organizations	0.096
organization_mentioned	0.082
person_location_mentioned	0.080
person_organization_mentioned	0.079
all_mentioned	0.067
n_persons	0.055
person_mentioned	0.023

Analyzing Table 5.2, we can see that the most important features were the number of locations and the respective boolean “location_mentioned". This indicates that the presence of locations in a text is probably important for the detection of relevance, which is corraborated by the high correlation between these attributes and the relevance, discussed in section 5.3.2.

The number of entities and the boolean feature “none_mentioned" scored immediately after. Since there was a high negative correlation between the absence of entities and the relevance of a post, it is likely that this feature could be very important for detecting “not relevant" posts.

5.4 Predictive Analysis

5.4.1 Methodology

5.4.1.1 Feature selection

In most of the experiments, the subsets of features were manually chosen, based on our intuition.

That being said, the subsets chosen were as follows. Our first subset of features consisted of the internal named entity-based features only, i.e., the 4 basic numerical features (number of persons, number of locations, number of organizations, number of total entities - represented in the experimental results as “1-4").

Then we added the dichotomous features (5-12) to the first subset, to see if using boolean features deduced from the numerical features would bring any benefit to the models.

The third subset was the previous one plus the externally obtained entity-features, based on The Guardian's mentions, as explained in 5.2.3.2, to see if that improved the models further.

We then repeated the above subsets but including word vectors as features as well.

Besides manual "feature selection" approaches, we also experimented Forward and Backward feature selection and compared those selections with our own.

5.4.1.2 Models creation and evaluation

In the following experiments, different models were trained using different machine learning algorithms. For each algorithm, the whole dataset was used for training and testing, using 10-fold cross-validation.

In order to fairly compare different models, we used the same seed for cross-validation and for the algorithms (if applicable).

For each model we will be reporting the Precision and Recall, F1-score (for the class "relevant") and AUC.

5.4.2 Experimental results

5.4.2.1 Features evaluation

In this subsection we will analyze the importance of each feature, by comparing results using different subsets of features and ranking their importance according to different metrics, in multiple scenarios.

In tables 5.3 and 5.4, the results of 14 different setups are presented. The results presented on 5.3 were obtained using a Random Forest (detailed information on the hyperparameters used can be consulted in appendix B.2.1) . The results presented on table 5.4 were obtained using a Naive Bayes classifier with Laplace correction.

Both algorithms were analyzed for 6 different subsets of features.

Entity-based features

Our first experiment consisted in using the 4 numerical features only (number of persons, locations, organizations and total sum of entities). The results were the lowest in terms of AUC for the random forest model, that also being the case for the Naive Bayes classifier.

The boolean features of mentioned entities (5-12) were mostly redundant: for example, the boolean feature "persons_mentioned" (ID 5 in table 5.1) adds no new information, given that we could deduce that from the feature "number of persons" being 0 or greater than 0.

However, both classifiers showed to learn better with these features included than without them: both the AUC and F1-score were higher using the second subset of features (1-12) when compared to using the first subset of features (1-4).

Table 5.3: Random Forest results

Random Forest on dataset 1 with Ensemble2-entities features				
Features	Precision	Recall	$F1$	AUC
1-4	83.22%	38.62%	52.75%	73.0%
1-12	82.97%	40.40%	54.33%	73.2%
1-19	83.22%	37.30%	51.49%	73.4%
20	97.62%	48.39%	64.70%	91.2%
1-12 $\cup$ 20	98.12%	52.51%	68.40%	92.6%
All	98.39%	51.41%	67.53%	92.5%

In the third experiment we added 7 features (13-19) on top of the original ones (1-12). These features were statistics provided from the newspaper *The Guardian*, and our results show that these features improved the Random Forest model in terms of AUC (by only 0.2%), but on the other hand reduced its F1-score. In the Naive Bayes case, both AUC and F1-score were lower when these features were added. Looking at the precision and recall for the class “relevant", present in table 5.4, we can see that these news-worthiness-based features made our models more precise, but decreased their recall.

In table 5.5, it is possible to see the feature weight values by Tree Importance, which uses a random forest model (with the same hyperparameters used for previous experiments, detailed in B.2.1) to extract the importance of each feature. The definition of such importance, from the official RapidMiner documentation [Wei], is as follows:

> “This weighting schema will use a given random forest to extract the implicit importance of the used attributes. Therefore each node of each tree is visited and the benefit created by the respective split is retrieved. This benefit is summed per attribute, that had been used for the split. The mean benefit over all trees is used as importance."

The results were similar to the importances analyzed in Table 5.2 for the top ranked features. However, the feature “n_organizations" was more important than the feature “none_mentioned".

Word-embeddings-based features

As previously mentioned, we wanted to not only evaluate entities as features for relevance detection, but to compare it with other state-of-the-art methodologies, such as word embeddings. The next three setups consisted of word embeddings only, word embeddings plus our entity-based features, and word embeddings, our entity-based features and the news-worthiness features.

Word embeddings achieved a significantly higher performance in every metric than our previous experiments with entities alone. This was actually expected, since word embeddings use all

Table 5.4: Naive Bayes results

Naive Bayes on dataset 1 with Ensemble2-entities features				
Features	Precision	Recall	$F1$-score	AUC
1-4	76.67%	57.44%	65.67%	72.7%
1-12	76.67%	57.44%	65.67%	73.0%
1-19	77.24%	42.58%	54.80%	72.7%
20	87.70%	72.16%	79.21%	91.3%
1-12 ∪ 20	87.97%	78.26%	82.83%	92.7%
All	88.26%	74.95%	81.06%	91.8%

the words present in the text, while our statistics about entities ignore most of the text. The interesting step, however, was to see if adding our entity-based features could improve the performance of word embeddings, which proved to be the case. Our high-level entity-based features led to an improvement of 2.1% in terms of AUC for the Random Forest model, and 1.7% in terms of AUC for the Naive Bayes model, as we can see in Tables 5.3 and 5.4.

Finally, our last experiment included every feature from Table 5.1. This time, adding the newsworthiness features did not improve our models in either case, and even reduced the performance of both classifiers in terms of AUC.

We can say that, in general, for both of the algorithms, the more features used, the better the performances of our classifiers.

In every of the above experiments, the AUC scores were higher than F1-scores.

While the AUC score takes into consideration the whole range of precision/recall trade-offs (thresholds), the F1 score considers only one specific precision and recall pair, in this case 0.5.

This indicates that the threshold of 0.5 was not the best one for the different models, and that there were different thresholds with better performance.

Automatic feature selection

In table 5.6 we present the results of Forward and Backward selection for the Naive Bayes algorithm. We chose this algorithm since it was the fastest among all the experimented algorithms. We also chose 2 subsets of features for this experiment: first using all the features available, including word embeddings, and then using only the internal named-entity features.

It is possible to see that both methods (forward and backward selection) worked better than our subsets of features. While this might seem like an obvious result, it is important to note that both these feature selection methods are greedy approaches, and therefore it could have been the case that the feature selection methods performed worse than our manually chosen subsets.

Table 5.5: Feature weights by Tree Importance

Features	Weight
location_mentioned	0.041
n_locations	0.032
n_entities	0.023
n_organizations	0.019
none_mentioned	0.016
n_persons	0.011
person_organization_mentioned	0.009
location_organization_mentioned	0.006
organization_mentioned	0.006

With forward selection, there was an improvement of 2.9% AUC, using all the features as input - in this case the algorithm only used the feature “none_mentioned" and 9 word vectors, excluding the remaining features. When using only the internal entity-based features, there was an improvement of 0.3% AUC, and the features selected were the n_locations, n_entities, persons_mentioned, locations_mentioned, organization_mentioned, person_organization_mentioned ,and location_organization_mentioned.

Using backward elimination, the improvement was only of 1.3% in terms of AUC, when using all the features as input - the external entity-based features were discarded, except for “persons_news_worthiness"; For the internal entity-based features, there was an improvement of 0.4%, and the features selected in this case were the n_locations, n_organizations, location_mentioned, person_location_mentioned, location_organization_mentioned, all_entities and persons_news_wo rthiness.

Table 5.6: Feature Selection

Feature selection experiments						
Method	Input features	Selection	Precision	Recall	$F1$-score	AUC
Forward	All	12 ∪ 9 word vectors	89.67%	80.06%	84.59%	94.7%
Forward	1-12	2,4,5,6,7,9,10	78.38%	50.45%	61.38%	73.3%
Backward	All	All -{14-19}	88.12%	78.85%	83.22%	93.2%
Backward	1-12	2,3,6,8,10,11,13	79.40%	51.46%	62.44%	73.4%

5.4.2.2 Algorithms evaluation

In section 5.4.2.1 two different ML algorithms were used, with the sole purpose of verifying if our conclusions about subsets of features depended heavily on the algorithm used or not.

In this section, however, our goal is to compare the performance of different ML algorithms on the task of relevance detection. In our comparisons we will use the performance metrics AUC and F1.

For the following experiments, we chose to use as features only the 4 numerical features (number of persons, number of locations, number of organizations, and number of entities). We decided to exclude the boolean features to maximize the number of algorithms used (since some of them allowed only numerical inputs). The news-worthiness features were also ignored, mainly due to the bad results in previous experiments. Finally, we did not use word embeddings since the goal was to isolate the entity-based features understand that way which algorithms worked better for them, without word embeddings likely interference.

Seven ML algorithms were experimented: Naive Bayes, Random Forest, Logistic Regression, SVM, Neural Network, Deep Learning and Decision Tree. The results (Precision, Recall, F1 and AUC) can be seen in Table 5.7. Specific details about the hyperparameters used for each algorithm can be seen in appendix B.2.1.

Table 5.7: Algorithms' performances comparison with numerical entity features

Algorithm	Precision	Recall	F1	AUC
Naive Bayes	76.67%	57.44%	65.67%	72.7%
Random Forest	83.13%	38.99%	53.07%	73.0%
Logistic Regression	76.67%	57.44%	65.67%	73.4%
SVM	77.69%	30.11%	40.12%	68.2%
Neural Network	77.85%	53.71%	63.49%	73.3%
Deep Learning	62.39%	55.12%	53.98%	57.9%
Decision Tree	83.24%	38.62%	52.75%	66.3%

5.4.2.3 External dataset validation

As mentioned in 5.2.2, we used an external dataset to validate our results.

It is possible to see in Table 5.8 that all validation results were worse than the original cross-validation results.

However, for the setups using entity-based features, the performances were still acceptable both in terms of F1 and AUC. For the subset of features 1-4, the F1-score dropped only 2.02% and the AUC 4.8%. For the second subset of features, 1-12, the difference was even smaller for AUC, with a drop of 4.1%.

Table 5.8: Original cross-validation results versus validation set results (Naive Bayes)

Original Results VS Validation Results				
Features	Original F1	Validation F1	Original AUC	Validation AUC
1-4	65.67%	63.65%	72.7%	67.9%
1-12	65.67%	63.65%	73.0%	68.9%
1-19	54.80%	55.41%	72.7%	67.7%
20	79.21	62.04%	91.3%	66.3%
1-12 ∪ 20	82.83%	67.36%	92.7%	73.4%
All	81.06%	68.80%	91.8%	73.5%

On the other hand, the setups using word embeddings revealed to have major drops in performance: for word embeddings alone, the F1 dropped 17.17% and the AUC 25%. Adding or entity-features, the gaps in performance were already slightly lower: 15.47% for F1 and 19.3% for AUC. With all features included, the differences were even smaller, with the F1 dropping 12.26% and AUC 18.3%.

Overall, despite the predictable drop of performance in the validation set, one could say that all the models were robust enough to still be useful in a significantly different dataset, from a different temporal window and from different users and topics.

While word embeddings learn the words present in the text directly, being therefore very susceptible to overfitting the model to the data, our entity-based features are more abstract and high-level, that being the likely reason of the respective models' robustness. We have witnessed significantly less difference in validation results using entity-based features only, and also that adding those features to word embeddings resulted in smaller performance differences between the two datasets.

5.4.2.4 Ensemble vs Stanford NER

All of the previous experiments were performed using for the entity extraction our Ensemble implementation using protocol 2. From our perspective it would be interesting, however, to see if our improved NER system had any impact on the relevance detection.

Looking at Table 5.9, it is possible to see the F1 and AUC results for both the Ensemble entities and the Stanford entities.

In terms of F1, the ensemble achieved higher values in all subsets of features, except for the third subset, the one that included news-worthiness from *The Guardian*.

On the other hand, in terms of AUC , the results were less consistent: for 2 subsets of features our Ensemble achieved higher score, and for the other 3 subsets the Stanford features were actually better.

Table 5.9: Ensemble Results versus Stanford Results using Naive Bayes

Ensemble Results VS Stanford Results				
Features	Ensemble F1	Stanford F1	Ensemble AUC	Stanford AUC
1-4	65.67%	58.81%	72.7%	74.4%
1-12	65.67%	64.04%	73.0%	74.5%
1-19	54.80%	62.16%	72.7%	75.5%
1-12 ∪ 20	82.83%	81.79%	92.7%	92.3%
All	81.06%	80.96%	91.8%	92.0%

These results were therefore not very conclusive. The ensemble features did not improve the results of relevance detection clearly, despite the fact that our previous experiments had shown the protocol 2 was better that CoreNLP by a large margin on the detection of the 3 entities (however, that margin was not so large for the entity type person, the most important in relevance detection).

5.5 Conclusions

Based on the previous comparisons, and respective results, the main conclusions of this study were as follows:

- **RQ1 answer**: as extensively explained in the previous chapter, there are multiple automatic systems for named entity extraction. We were able to improve that accuracy by creating and Ensemble of tools. That system was then used to provide features for our dataset
- **RQ2 answer**: entities mentioned in a social media post can provide useful features for a relevance detection prediction model
- **RQ3 answer**: simple statistics about the number of entities mentioned in a post were enough to achieve a performance up to 73.04% of AUC, without using the text of the entities
- **RQ4 answer**: in terms of the importance of the features used for relevance detection (RQ3), we have witnessed that locations were more important than organizations or persons, and that the absence of entities was very important to detect non-relevant posts
- **RQ5 answer**: our entity-based features were also useful when added on top of word embeddings-features, showing to improve the performance of our classifiers up to 1.4%.
- **RQ6 answer**: the models showed, as expected, losses in performance when tested on an independent validation dataset. However, this losses were significantly higher on word-embeddings based features than in entity-based features, corroborating the idea that entity-based features are more general and less prone to overfitting.

Chapter 6

Conclusions

6.1 Synthesis

The main focus of this thesis was to study the viability of using named entities mentioned in social media posts as features for training a machine learning model to predict the posts' relevance. Therefore, the success of this analysis depended heavily on the success of extracting named entities in the first place.

Named entity recognition is a particularly difficult task in social media texts, since they carry specific characteristics, such as spelling errors or incorrect casing, as a result of their informal nature. Different approaches have been attempted to improve Named Entity Recognition in general, and also on this specific type of text (social media), but so far there is not a single method or tool that outperformed all others and that is widely recognized as the best NER system. In that regard, our first task was to find the most widely known NER tools, study and try them experimentally on our own proprietary datasets, and find out which one worked out better for our purposes.

While many approaches have been proposed in literature, most of them lack an out-of-the-box tool, easy to set up and reproduce the results. As implementing a system from scratch is very time consuming and not the main goal of this study, we opted to try four different out-of-the-box systems, namely Stanford NER, Twitter NLP tools, GATE with TwitIE and OpenNLP.

The first conclusion of this analysis was that, once again, there was not a single tool that outperformed all others in every entity type (even though GATE performed overall better than any other toolkit across multiple datasets). In order to try to make full use of these differences between toolkits, we decided to create an *Ensemble of toolkits*, first by using manually defined voting rules, and later by using machine learning algorithms to predict the final entity for each token, using as features the inputs of each tool.

Both of these approaches achieved promising results: the results showed that using an ensemble of toolkits can improve the recognition of specific entity types, depending on the criteria used for the voting, and even the overall performance average of the entity types PERSON, LOCATION and ORGANIZATION. Besides the scientific contribution of these findings, we developed an easy to set up system that combines 4 state-of-the-art toolkits with improved performance over each tool individually. This way, anyone looking for an out-of-the-box NER system could use our ensemble for their own works.

With that being said, we proceeded to the second and main part of this study — the analysis of the importance of named entities for relevance detection.

The first task was to generate features from the entities extracted. We created basic statistics based on the number of mentions in each post (number of persons in the post, for instance), and statistics obtained from *The Guardian*'s open API, which returned the number of times a specific entity had been mentioned in the news in a given span of time, giving a sense of "news-worthiness" for each entity.

We started by picking one algorithm (random forest) and testing it for our features based on the extraction of the ensemble tool versus the features based on the extraction of the Stanford toolkit. We concluded that our improved ensemble had a slight impact on the F1-scores, despite the inconclusive results in terms of AUC.

After that experiment, we chose the toolkit that provided the features with best relevance detection performance. This was a greedy choice, and probably not optimal, but testing every combination of toolkits, datasets, algorithms, hyperparameters, would not be feasible. Therefore we opted for this approach. These features were used as inputs for training and testing our prediction models. We tried 7 machine learning algorithms (Naive Bayes, Random Forest, kNN, Logistic Regression, SVM and Neural Network), using static hyperparameters (no tuning). The best results were achieved by Logistic Regression, for the 4 basic entity features. Therefore, we can conclude that these entities were enough to predict relevance up to 73.4% AUC, when used in isolation.

Then, to better understand the impact of each feature, we ranked them by various criteria, such as "Information Gain Ratio" using only entity-based features. The results showed that locations were the most important entity type for relevance detection. The correlation between our features and the label "Relevance" showed that the number of entities was the highest correlated with the "relevant" label, while no named entities was the highest correlated with the "not relevant" label.

We also experimented using different subsets of features for the same ML algorithm, to analyze the importance of each feature subset in the task of predicting relevance. The general tendency was that the more features used, the better the performance of our models. In addition, we experimented the use of word embeddings as features for the relevance classifier, in isolation, and later combined with our entity-based features. Adding our features to the word embeddings showed to improve the performance of the classifiers.

Lastly, to further investigate the versatility of the developed models, we used an external dataset for validation. The results showed that there was no major overfitting to the original dataset, and that word-embeddings performed significantly worse on the external dataset while entity-based features managed to obtain not very different results.

Our final conclusion is that information about named entities mentioned in social media posts can help in the task of relevance detection, as the presence of some entity types, such as location, showed to be important to detect relevant posts, and the complete absence of entities important to detect irrelevant posts. Entities alone did not achieve better results than other previous methods tested in our project, such as word embeddings. That is not very surprising, given that entity recognition in social media is very hard and that state-of-the-art performances are far from 100%. That being said, having machine learning models learning from features obtained by machine learning themselves necessarily leads to a significant error accumulation. However, the very abstract nature of entity-based features showed to be significantly more robust across different datasets than word embeddings, and were able to improve previous models (word embeddings) using this type of features.

6.2 Contributions

The main contributions of this thesis for the computer science community were the following:

- Publication of the scientific paper “The complementary nature of different NLP toolkits for Named Entity Recognition in social media" to the “18th EPIA Conference on Artificial Intelligence":
  - Analysis of 4 state-of-the-art NER tools (Stanford NER, OpenNLP, Twitter NLP tools and GATE with TwitIE) on social media datasets
  - Creation of an Ensemble of toolkits, with a combined performance superior to each individual tool.
- Creation of an automatic named-entity feature extraction system
- Study on the viability of such features in the task of relevance detection
- Conclusions on the importance of specific named entity types for relevance assessment.

6.3 Future work

As future work, additions could be made to improve the Ensemble of toolkits. More toolkits could be added to the system, such as NLTK or others. It would also be interesting to experiment more refined protocols — for example with weighted voting based on the performance of each toolkit for a given entity type — and machine learning algorithms, in the latter case by tuning the hyperparameters. Other entity types could also be explored, such as Dates, which are an important element of news articles and therefore the mention of such entities could help in the detection of relevance, and also other less common entity types such as PRODUCT, COMPANY, BAND, MOVIE, etc.

In terms of the results, a deeper analysis could be conducted in the future in order to better understand the behaviours observed in each toolkit, as well as the differences across corpora. Statistical tests would also be interesting to check if improvements between tools are statistically significant or not.

For the machine learning algorithms, more complex features and hyperparameters could be tried and analyzed. It would also be interesting to apply the ML approach to different datasets and compare the results.

In terms of the relevance detection system, more entity types could be used as well. In addition, more models could be trained using more feature subsets, feature selection methods could be improved (using for instance embedded methods instead of wrapper methods) and the hyperparameters for each algorithm could be tuned for optimal performance.

References

[AL13] Samet Atdağ and Vincent Labatut. A comparison of named entity recognition tools applied to biographical texts. In *Systems and Computer Science (ICSCS), 2013 2nd International Conference on*, pages 228–233. IEEE, 2013.

[ARS13] Rami Al-Rfou and Steven Skiena. Speedread: A fast named entity recognition pipeline. *arXiv preprint arXiv:1301.2857*, 2013.

[AZ12] Charu C Aggarwal and ChengXiang Zhai. *Mining text data*. Springer Science & Business Media, 2012.

[BDF$^+$13] Kalina Bontcheva, Leon Derczynski, Adam Funk, Mark A Greenwood, Diana Maynard, and Niraj Aswani. Twitie: An open-source information extraction pipeline for microblog text. In *RANLP*, pages 83–90, 2013.

[BDMH$^+$15] Timothy Baldwin, Marie Catherine De Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text (WNUT 2015), Beijing, China*, 2015.

[BEP$^+$08] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

[BG13] Marcel Broersma and Todd Graham. Twitter as a news source: How dutch and british newspapers used tweets in their news coverage, 2007–2011. *Journalism Practice*, 7(4):446–464, 2013.

[BKL09] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[CBVR$^+$13] Amparo Elizabeth Cano Basave, Andrea Varga, Matthew Rowe, Milan Stankovic, and Aba-Sah Dadzie. Making sense of microposts (# msm2013) concept extraction challenge. 2013.

[CFL13] Alexander Clark, Chris Fox, and Shalom Lappin. *The handbook of computational linguistics and natural language processing*. John Wiley & Sons, 2013.

[CMBT02] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. A framework and graphical development environment for robust nlp tools and applications. In *ACL*, pages 168–175, 2002.

[Cor16] Stanford CoreNLP. a suite of core nlp tools. *URL http://nlp. stanford. edu/software/corenlp. shtml*, 2016.

[DMR$^+$15] Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, 51(2):32–49, 2015.

[FGM05] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[For07] George Forman. Feature selection for text classification. *Computational methods of feature selection*, 1944355797, 2007.

[Fou] The Apache Software Foundation. Apache opennlp. https://opennlp.apache.org/. Accessed: 2017-04-21.

[FSF16] Alvaro Figueira, Miguel Sandim, and Paula Fortuna. An approach to relevancy detection: Contributions to the automatic detection of relevance in social networks. In *New Advances in Information Systems and Technologies*, pages 89–99. Springer, 2016.

[GAPGC$^+$13] Eni Mustafaraj Markus Strohmaier Harald Schoen Gayo-Avello, Panagiotis Takis Metaxas, Daniel Peter Gloor, Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Predicting information credibility in time-sensitive social media. *Internet Research*, 23(5):560–588, 2013.

[GAT] Gate. `https://gate.ac.uk/`. Accessed: 2017-04-21.

[GE03] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[GE06] Isabelle Guyon and André Elisseeff. An introduction to feature extraction. In *Feature extraction*, pages 1–25. Springer, 2006.

[HCC11] Lichan Hong, Gregorio Convertino, and Ed H Chi. Language matters in twitter: A large scale study. In *ICWSM*, 2011.

[JBL16] Ridong Jiang, Rafael E Banchs, and Haizhou Li. Evaluating and combining named entity recognition systems. *ACL 2016*, page 21, 2016.

[Ker16] Sara Keretna. *Named Entity Recognition in Unstructured Text*. PhD thesis, Deakin University, Australia, 2016.

[KLPM10] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

[KQ13] Mehnaz Khan and SMK Quadri. Effect of using filter based feature selection on performance of machine learners using different datasets. *BVICAM's International Journal of Information Technology*, 5:597–603, 2013.

[LD11] L Ladha and T Deepa. Feature selection methods and algorithms. *International journal on computer science and engineering*, 3(5):1787–1797, 2011.

[LM07] Huan Liu and Hiroshi Motoda. *Computational methods of feature selection.* CRC Press, 2007.

[LSTO10] Gustavo Laboreiro, Luís Sarmento, Jorge Teixeira, and Eugénio Oliveira. Tokenizing micro-blogging messages using a text classification approach. In *Proceedings of the fourth workshop on Analytics for noisy unstructured text data*, pages 81–88. ACM, 2010.

[LZWZ11] Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 359–367. Association for Computational Linguistics, 2011.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[MCG15] Carlos Martin, David Corney, and Ayse Goker. Mining newsworthy topics from social media. In *Advances in Social Media Analysis*, pages 21–43. Springer, 2015.

[MMS93] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[MSB$^+$14] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[MSC$^+$13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[MWC05] Einat Minkov, Richard C Wang, and William W Cohen. Extracting personal names from email: Applying named entity recognition to informal text. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 443–450. Association for Computational Linguistics, 2005.

[NBG15] Kamel Nebhi, Kalina Bontcheva, and Genevieve Gorrell. Restoring capitalization in# tweets. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1111–1115. ACM, 2015.

[PGOOA16] Alexandre Pinto, Hugo Gonçalo Oliveira, and Ana Oliveira Alves. Comparing the performance of different nlp toolkits in formal and social media text. In *OASIcs-OpenAccess Series in Informatics*, volume 51. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[RBBL12] Kepa Joseba Rodriquez, Mike Bryant, Tobias Blanke, and Magdalena Luszczynska. Comparison of named entity recognition tools for raw ocr text. In *KONVENS*, pages 410–414, 2012.

[RCE$^+$11] Alan Ritter, Sam Clark, Oren Etzioni, et al. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics, 2011.

[RM99] Lance A Ramshaw and Mitchell P Marcus. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer, 1999.

[SE13] Sriparna Saha and Asif Ekbal. Combining multiple classifiers using vote based classifier ensemble technique for named entity recognition. *Data & Knowledge Engineering*, 85:15–39, 2013.

[Seb02] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[SMABTS07] Noelia Sánchez-Maroño, Amparo Alonso-Betanzos, and María Tombilla-Sanromán. Filter methods for feature selection–a comparative study. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 178–187. Springer, 2007.

[SN14] René Speck and Axel-Cyrille Ngonga Ngomo. Ensemble learning for named entity recognition. In *International Semantic Web Conference*, pages 519–534. Springer, 2014.

[TKSDM03] Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.

[TS12] Maksim Tkachenko and Andrey Simanovsky. Named entity recognition: Exploring features. In *KONVENS*, pages 118–127, 2012.

[WA86] Donald Walker and Robert Amsler. The use of machine-readable dictionaries in sublanguage analysis. *Analyzing Language in Restricted Domains*, pages 69–83, 1986.

[Wei] Weight by tree importance - rapidminer documentation. https://docs.rapidminer.com/studio/operators/modeling/feature_weights/weight_by_forest.html. Accessed: 2017-06-17.

[WJTH06] Chia-Wei Wu, Shyh-Yi Jan, Richard Tzong-Han Tsai, and Wen-Lian Hsu. On using ensemble methods for chinese named entity recognition. In *Proceedings of the 5th SIGHAN Workshop on Chinese Language Processing*, pages 142–145, 2006.

[WNC03] Dekai Wu, Grace Ngai, and Marine Carpuat. A stacked, voted, stacked model for named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 200–203. Association for Computational Linguistics, 2003.

Appendix A

Chapter 4 Extended results

Table A.1: Recall scores on Dataset 2

	PERSON	LOCATION	ORGANIZATION	Avg.
CoreNLP	71.84	20.60	13.04	35.16
TwitIE	70.33	40.07	45.89	52.09
TwitterNLP	39.31	22.28	32.37	31.32
openNLP	34.49	25.28	12.56	24.11
$Ensemble_1$	62.65	27.90	36.23	42.26
$Ensemble_2$	64.61	30.90	37.20	44.24

Table A.2: Precision scores on Dataset 2

	PERSON	LOCATION	ORGANIZATION	Avg.
CoreNLP	46.72	76.92	42.86	55.5
TwitIE	52.24	60.28	32.76	48.43
TwitterNLP	80.31	80.41	74.44	78.39
openNLP	57.11	55.79	4.47	39.12
$Ensemble_1$	80.78	80.54	51.02	70.78
$Ensemble_2$	77.44	79.71	47.53	68.23

Table A.3: Recall scores on Dataset 3

	PERSON	LOCATION	ORGANIZATION	Avg.
CoreNLP	81.24	40.44	16.69	46.12
TwitIE	76.35	63.68	43.34	61.12
TwitterNLP	39.74	30.87	9.04	26.55
openNLP	47.41	36.68	28.39	37.49
$Ensemble_1$	71.85	48.31	29.32	49.83
$Ensemble_2$	76.02	54.5	33.49	54.67

Table A.4: Precision scores on Dataset 3

	PERSON	LOCATION	ORGANIZATION	Avg.
CoreNLP	60.27	82.06	72	71.44
TwitIE	77.79	72.85	44.58	65.07
TwitterNLP	89.51	65.22	77.23	77.32
openNLP	66.65	68.09	23.09	52.61
$Ensemble_1$	90.06	87.31	70.28	82.55
$Ensemble_2$	89.85	84.77	69.81	81.48

Appendix B

Detailed specifications of experimental setups

B.1 Ensemble NER

B.1.1 Naïve Bayes classifier:

For this algorithm the following parameters were used:

- Laplace correction

B.1.2 Random Forest classifier:

For this algorithm the following parameters were used:

- Number of trees: 10
- Criterion: Gain ratio
- Maximal depth: 20
- Pruning with a confidence of 0.25
- Pre-pruning
- Minimal gain: 0.1
- Minimal leaf size: 2
- Minimal size for split: 4
- Number of pre-pruning alternatives: 3
- Voting strategy: confidence vote

B.1.3 k-nearest neighbors classifier (k-NN):

For this algorithm the following parameters were used:

- k of 3 and k of 10
- Mixed Euclidean Distance

B.1.4 Neural Network classifier:

For this algorithm the following parameters were used:

- No hidden layers
- 500 training cycles
- Learning rate: 0.3
- Momentum: 0.2
- Shuffling and normalization
- Error epsilon: 1.5ε-5

B.2 Relevance Detection

B.2.1 Random Forest classifier:

This classifier was built using the "Neural Net" RapidMiner operator. For further details consult the official documentation. For this algorithm the following parameters were used:

- 200 trees
- Criterion: information gain
- maximal depth: 20
- Minimal gain: 0.1
- Minimal leaf size: 2
- Minimal size for split: 4
- Number of prepruning alternatives: 3
- Voting strategy: confidence vote

B.2.2 Logistic Regression classifier:

This classifier was built using the "Neural Net" RapidMiner operator. For further details consult the official documentation. For this algorithm the following parameters were used:

- solver: AUTO
- reproducible: disabled
- use regularization: disabled
- standardize: enabled
- non-negative coefficients: disabled
- add intercept: enabled
- compute p-valued: enabled
- remove collinear columns: enabled
- Missing values handling: mean imputation
- max iterations: 0
- max runtime seconds: 0

B.2.3 SVM classifier:

This classifier was built using the "Neural Net" RapidMiner operator. For further details consult the official documentation. For this algorithm the following parameters were used:

- Kernel type: dot
- Kernel cache: 200
- C: 0.0
- Convergence epsilon: 0.001
- Max iterations: 100000
- Scale: enabled
- L pos: 1.0
- L neg: 1.0
- epsilon: 0.0
- epsilon plus: 0.0

- epsilon minus: 0.0
- balance cost: disabled
- quadratic loss pos: disabled
- quadratic loss neg: disabled

B.2.4 Neural Network classifier:

This classifier was built using the "Neural Net" RapidMiner operator. For further details consult the official documentationofficial documentation. For this algorithm the following parameters were used:

- 1 hidden layer: 100 neurons
- Training cycles: 500
- Learning rate: 0.3
- Momentum: 0.2
- Decay: disabled
- Shuffle: enabled
- Normalize: enabled
- Error epsilon: $1.0e^{-5}$

B.2.5 Deep Learning classifier:

This classifier was built using the "Deep Learning" RapidMiner operator. For further details consult the official documentation. For this algorithm the following parameters were used:

- Activation: Rectifier
- 2 hidden layers of 50 neurons each
- Epochs: 10
- Compute variable importances: disabled
- Train samples per iterations: auto-tuning
- Adaptive rate: enabled
- Epsilon: $1.0e^{-8}$
- rho: 0.99

- Standardize data: enabled
- L1: $1.0e^{-5}$
- L2: 0.0
- max w2: 10.0
- loss function: automatic
- distribution function: AUTO
- early stopping: disabled
- missing values handling: Mean Imputation

B.2.6 Decision Tree classifier:

This classifier was built using the "Decision Tree" RapidMiner operator. For further details consult the official documentation. For this algorithm the following parameters were used:

- Criterion: Gain ratio
- maximal depth: 20
- apply pruning: enabled
- confidence: 0.25
- apply prepruning: enabled
- Minimal gain: 0.1
- Minimal leaf size: 2
- Minimal size for split: 4
- Number of prepruning alternatives: 3