NTNU
Norwegian University of
Science and Technology

Multilingual News Article Classification

Patrick L Skjennum

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Jon Atle Gulla, IDI
Co-supervisor: Jon Espen Ingvaldsen, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

ABSTRACT

News is an ever-growing and global resource, reliant on robust distribution networks to spread information. This thesis investigates how exploiting semantic, contextual and ontological information may form a basis for a language-independent news article classification system.

In light of the above, a scalable multi-label news article classification system, based exclusively on extracted DBpedia entities, and a predetermined standardized set of fixed-size *IPTC Media Topic* categories, is presented. The proposed system includes an ensemble of n-binary multinominal classifiers, comprised of both traditional Naïve Bayes and several sophisticated artificial neural networks – all trained on 1.8 million news articles, spanning twenty years of content from *The New York Times*.

Through a series of experiments, this thesis provides evidence that a reliable language-independent news article classifier is plausible – achieving a macro-averaged F-score of 91% in categories like *sport,* and an overall F-score of 49% for the whole system. Furthermore, the results show that utilizing pre-trained word embeddings like Word2Vec over the traditional *Bag-of-Words* approach for feature representation, provides both reduced training time and comparable classification quality. Also included in the experiments are several studies exploring how article length, incorporation of ontologically related supertypes, and moving through time, affects the classification quality of news articles. Among the most central findings is that article length is positively correlated with F-score up until a length of 600 words, at which point the F-score stabilizes.

Finally, the thesis presents a thorough evaluation comparing traditional machine learning to the *state-of-the-art* in deep learning for the news article domain, both from a theoretical and practical standpoint – ultimately concluding that replacing conventional and well-performing machine methods with deep learning is not necessarily the right solution in simple problem domains.

SAMMENDRAG

Nyheter er en voksende, og global ressurs, avhengig av robuste distribusjonsnettverk for å spre informasjon. Denne avhandlingen undersøker hvordan semantisk, kontekstuell og ontologisk informasjon kan utnyttes for å danne et grunnlag for et språkuavhengig nyhetskategoriseringssystem.

I lys av det ovenstående, presenterer dette studiet et skalerbart multikategorisk nyhetskategoriseringssytem, basert utelukkende på uthentede DBpedia-entiteter og et forhåndsbestemt og standardisert sett med IPTC Media Topic kategorier. Det foreslåtte systemet er satt sammen av n-binære multinominale klassifikatorer, deriblant den tradisjonelle Naive Bayes klassifikatoren, og flere sofistikerte kunstige neurale nettverk – alle trent opp med et datagrunnlag på 1,8 millioner nyhetsartikler bestående av 20 år med innhold fra *The New York Times*.

Gjennom en serie eksperimenter viser denne avhandlingen at et pålitelig og språkuavhengig nyhetskategoriseringssytem er mulig – og oppnår en F-score med makrogjennomsnitt på 91% for enkelte kategorier som *sport* og en samlet verdi på 49% for hele systemet. Videre viser studiet at man ved å benytte forhåndstrente Word2Vec modeller som et alternativ til standard *bag-of-words* for å representere data, både gir redusert treningstid og sammenlignbar klassifiseringskvalitet. I tillegg inneholder oppgaven flere eksperimenter som undersøker hvordan artikkellengde, inkorporering av ontologiske supertyper, samt hvordan det å bevege seg i tid, påvirker klassifiseringskvaliteten av nyhetsartikler. Blant de mest sentrale funnene er observasjonen at artikkellengde er positivt korrelert med F-score inntil en lengde på rundt 600 ord, hvor verdien så stabiliseres.

Også inkludert i oppgaven er en grundig gjennomgang av *state-of-the-art* innen *deep learning*, samt en studie rundt hvor godt disse klassifiseringsmetodene fungerer sammenlignet med tradisjonelle maskinlæringsmetoder når det kommer til klassifisering av nyhetsartikler. Løst fortalt konkluderer studiet med at deep learning ikke nødvendigvis alltid er den riktige løsning i enkle problemdomener.

PREFACE

This thesis is submitted to the *Norwegian University of Science and Technology* (NTNU) as a part of the fulfillment of a Master of Science degree in Computer Science.

The research is conducted at the *Department of Computer and Information Services* (IDI), and is supervised by Jon Espen Ingvaldsen, as a part of the *SmartMedia*[1] project administered by Prof. Jon Atle Gulla.

[1] *SmartMedia – https://www.ntnu.no/wiki/display/smartmedia/SmartMedia+Program*

Acknowledgements

First and foremost, I would like to express my gratitude towards my supervisor Jon Espen Ingvaldsen for providing me with constant and invaluable feedback throughout the work on the thesis. Not only has he been a huge motivator, but he has also been incredibly patient and resourceful, ultimately making this project a joy. I would also like to thank Prof. Jon Atle Gulla for enabling the project through the SmartMedia Program, and for providing me with excellent feedback when I needed it the most.

Furthermore, I would like to throw big thanks to Adam Gibson, Alex D. Black and Paul Dubs from the *Deeplearning4j* team, for giving me almost daily advice and help with both their software and generally configuring, training and working with artificial neural nets. The same thanks goes out to Justin du Coeur and Dennis Huo for helping me out whenever I had problems with Scala or Apache Spark.

I would also like to thank Arne Dag Fidjestøl for providing me with several servers, and incredible 24/7 tech support.

In addition, I would like to give special thanks to Evan Sandhaus from the *New York Times Research and Development Labs*, and the creator of the *New York Times corpus*, for providing me with the *New York Times to IPTC Media Topics* mapping. Without his support, this thesis would not have been possible.

For reading through the report and providing me with corrections and linguistic advice, I would like to thank my sister, Pia, and my awesome roomie Fredrik.

Contents

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

This initial chapter serves as an overall presentation of the problem domain and research context. Included in this chapter is the preparation and elaboration of important research questions, motivated by some of the major news related obstacles faced by society today. A roadmap for the project's structure and an outline for the research conducted, concludes the chapter.

1.1 BACKGROUND AND MOTIVATION

Ever since the dawn of time people have processed, stored and shared information about occurring events. Even before the spike of literacy in the modern world, *news* has been an integral part of society. During this period, news as a medium has played a major role in keeping the world united and up-to-date. Yet, what once was the responsibility of the common tongue and heavily dependent on travel & trade, is now encapsulating us in every direction – and overwhelmingly so. News has become an omnipresent, and universally shared resource, spanning national borders and cultural differences: Ranging from trivial gossip to war-zone status reports. As a consequence, people have become extremely reliant on search engines and organized broadcasters, to conceive and digest even just a glimpse of what is happening throughout the world.

At the same time, the number of hours an individual can devote to make sense of the ever-growing flow of news is limited. Taking this into account, an incredibly complex challenge emerges: How shall relevant news be directed to the right people, in accordance with their personal preferences?

The first, and arguably most important step in this process, is proper categorization. It simply does not matter whether the system is successful in determining a user's preference, if the news itself is mislabeled. For this reason, the remaining steps in this operation depend heavily on robust categorization procedures – without it, everything breaks apart.

For a long time, news categorization has been a manual task carried out by librarians and domain experts [1] [2]. Although recent research shows promising prospects for automated approaches, current solutions often focus on the simple analysis of textual content, rather than exploring the potential in incorporating conceptual relations, and semantic properties [3]. The reason for this approach has traditionally been motivated by limited processing power and mediocre training data [4].

However, in light of recent advancements in distributed computation and publicly available metadata, new opportunities unravel. This brings us down to the heart of the thesis, and the core motivational driver behind the project: *To explore the relevancy of semantic and conceptual metadata, to improve the task of automatically categorizing news articles in a language-independent and global domain.*

1.2 PROBLEM DESCRIPTION

Most media houses and news publishers assign category labels to their news articles. However, these category annotation practices are often not standardized. When aggregating news from multiple sources one can observe that publishers apply different labels for similar categories. Furthermore, the annotation process is often done at various levels of detail, with lacking precision, and sometimes even missing altogether.

The goal of this thesis is to investigate automated approaches for homogeneous news article classification, motivated by language agnostic principles and recent advancements in *deep learning*. This involves a study of available news categorization standards, textual extraction and classification techniques, as well as available datasets. Evaluation should be done both in terms of classification quality and in light of the underlying computational requirements.

In addition, the research conducted throughout this thesis should be backed by a prototypical implementation of a news article classification system. The implementation should make use of *state-of-the-art* analytical frameworks, like Apache Spark[2] – primarily to facilitate large-scale distributed computation and analysis.

1.3 RESEARCH CONTEXT

The thesis is conducted as a part of the *TDT4900 Master's Thesis in Computer Science*, in conjunction with the *Computer Science Master* program at the *Norwegian University of Science and Technology* (NTNU).

The core motivator behind this project is the *NTNU SmartMedia Program* organized by Prof. Jon Atle Gulla in collaboration with the *Department of Computer and Information Services* at NTNU. The SmartMedia Program revolves around optimizing the mapping between relevant news and attracted consumers. By exploiting semantic and geospatial exploration, the program aims to aid the media industry in navigating and making sense of the enormous amount of available news sources around the world. The ultimate goal of the program is to deliver a mobile and context-aware news experience based on deep understanding of textual content.

1.4 RESEARCH GOALS AND QUESTIONS

The research for this thesis is split into three main phases. The first phase focuses on researching, experimenting and challenging current work in areas of relevance – ranging from general text categorization to utilizing ontologies and semantic metadata to improve existing methods. The purpose of this phase is to create an overview of *the state-of-the-*

[2] *Apache Spark – http://spark.apache.org/*

art in news classification, and to obtain a robust and reliable set of prerequisites for establishing a theoretical basis for the later experiments.

The second phase emphasizes on analyzing, manipulating and otherwise preprocessing and combining various data sources. This is followed by an implementation of a framework for automated news classification. In the third and concluding phase, the focus shifts over to conducting a set of experiments, followed by a thorough analysis of the relevant findings and an evaluation of the results.

To act as a foundation for the thesis, and as a summary of the underlying goals, the following research questions have been defined:

RQ1 *How does deep learning compare to traditional machine learning techniques in news categorization in terms of classification quality?*

RQ2 *How does word embeddings compare to a bag-of-word approach in news categorization?*

RQ3 *How does the number of annotations and article lengths affect the accuracy of news categorization?*

RQ4 *How does moving through time affect news categorization performance, and what is the relevance of up-to-date training data?*

RQ5 *How does incorporating ontologically related supertypes affect the quality of news categorization?*

RQ6 *How does noun-based feature extraction of ontological data promote language agnostic news categorization?*

RQ7 *What are the important limitations or challenges of deep learning in news categorization, and under what circumstances are the additional complexities of deep learning justified?*

1.5 REPORT OUTLINE

The remaining parts of the report are structured as follows: Chapter 2 provides a theoretical overview of the most central techniques and concepts related to text classification, with emphasis on the news article domain. Included in this chapter is also a thorough review of prevalent techniques in *deep learning*. Related work and a brief look into the *state-of-the-art* is covered in Chapter 3. Next, Chapter 4 follows with a substantial analysis of the relevant datasets as well as a thorough elaboration of how the datasets were preprocessed and utilized. This analysis is continued in Chapter 5 which covers the practicalities regarding the choice of classifiers and the training phase. The conducted experiments and results are presented in Chapter 6. Chapter 7 concludes the thesis by addressing research questions and proposing future work.

2 BACKGROUND

The task of text categorization can be summarized as the process of labeling a textual document with one or more predefined categories [5]. A natural way to represent text categorization is as a sequential pipeline with three steps [6]: The first step being (i) *preprocessing*, followed by (ii) applying the *classification method* of choice, and finally (iii) *evaluating* the results and choosing a desirable classifier based on chosen evaluation measures. For the sake of clarity, *categorization* and *classification* are used interchangeably throughout this report.

The preprocessing step typically involves indexing and transforming the textual data into a numerical format suitable for further processing [6], and is perhaps the most crucial step in the pipeline. Without a reasonable representation of the documents, the classification mechanism will have an extremely hard time grouping, dividing, and making sense of similar entities, thus making the task of *categorization* very difficult. For this reason, careful thought has to go into the process of determining the most suited methods for every step in the pipeline.

The primary motive for this chapter is to present a systematic breakdown of the classification process. This is done by first considering a set of common challenges faced in the news domain today, followed by a brief introduction to the field of semantics. The remaining parts of the chapter are devoted to a thorough evaluation of central algorithms, measures, and mechanics used in practice – both traditionally and the *state-of-the-art*.

2.1 SEMANTICS AND ITS CHALLENGES

Before exploring the theoretical methodologies, one has to take a step back and consider some of the challenges commonly faced in the news domain. As proposed by *The New York Times* [7], these challenges may be summarized into four different groups:

DISAMBIGUATION *Is this story about Kristen Bell, Bell Labs or Taco Bell? There is no way to know for sure, by only examining the textual content.*

SUMMARIZATION *The article includes a quote from Archimedes, but the context is actually climate change rather than philosophy. Summarization is the task of concisely conveying what the article actually is about. Typically, through metadata.*

NORMALIZATION *NY Times, NYT, New York Times, The Times, all refer to the same entity. Can all of them safely be labeled "Newspaper, New York"?*

ONTOLOGY *One story is about a video game, another about a new TV-show. Can both be labeled as subcategories of "Entertainment"?*

One way to address the challenges introduced by the concepts above, is to advance the analytical process beyond traditional methods. In this study, the advancement revolves around exploring the potential in incorporating semantic analysis – a field of research dedicated to extracting the *meaning* from expressions, beyond the power of conventional content analysis.

2.1.1 SEMANTICS

Formally, *semantics* is defined as "the study of meaning"[3]. In linguistic terms, and for the purpose of this thesis, this definition is assumed to be analogous to the *message* communicated through a word or a sentence. This contrasts with *syntax,* which defines the concrete structure, and the actual words being used. The importance of this distinction may not be obvious at first glance, but consider the following sentences:

(i) *Bob only enjoys playing catch with his neighbor's friend, the family dog.*

(ii) *The neighbor's family dog enjoys playing catch with his only friend, Bob.*

While the sentences contain the exact same set of words, their semantic meanings are conflicting: The first illustrates that the only dog Bob would want to play catch with, is the neighbor's dog, whereas the latter implies that the neighbor's dog is in fact quite lonely.

Needless to say, ambiguities like these are bound to occur when categorizing text solely based on textual contents, rather than its *meaning.* Not only does this lead to improper classification, but it also adds a level of confusion to the classifier [3] [8].

Luckily, there is an entire related field of research in natural language processing dedicated to solving these issues, commonly referred to as *Word-sense disambiguation*[4] – that is, the task of extracting *sense* from words with multiple semantic meanings [8].

2.1.2 ONTOLOGIES

The exact meaning of the word *ontology* has for a long time been subject to many controversies, resulting in a variety of definitions[5]. However, the most commonly accepted definition in the field of machine learning tends to be *T. R. Gruber's* interpretation [9]:

> *An ontology is an explicit specification of a conceptualization.*

With other words, it is a formal definition of concepts and the relations between them.

---

[3] *Semantics definition – http://dictionary.reference.com/browse/semantics*

[4] *Word-sense disambiguation – https://en.wikipedia.org/wiki/Word-sense_disambiguation*

[5] *Ontology definition – http://dictionary.reference.com/browse/ontology*

Why is this useful? Consider the following headlines:

(i) *Best Barbecue tips for the summer*

(ii) *Delicious, healthy, and organic cookie dough*

Although both of the headlines certainly fit the *cooking* profile, the contents of the articles themselves may not necessarily have that much in common. While the first article may contain words like *chicken, pork, tender* and *charcoal*, the latter might include *baking soda, healthy, vegetable,* and *dairy*. Viewed from the perspective of a computer, there is no obvious way to link the two together, even though they are closely related in practice.

A possible solution to this matter is to look for relations in an ontology [3], like the one exemplified in Figure 2-1. By locating each of the relevant words, and traversing from below, one would quickly reveal the fact that both *vegetable* and *chicken* are children of *organic* and *consumable*. Thus, a relation is made.

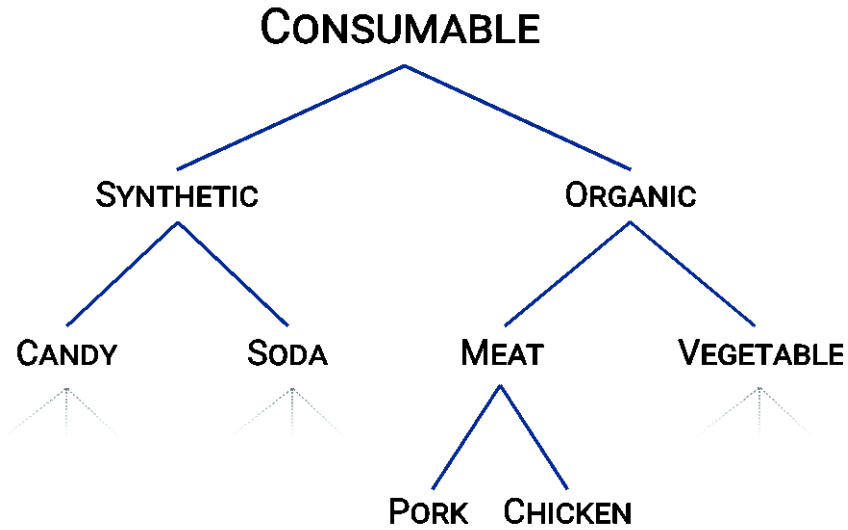


FIGURE 2-1 – EXAMPLE FOOD ONTOLOGY SHOWING "IS-A"-RELATIONSHIP

While the relation in the previous example only contains a simple *is-a*-relationship, it illustrates an important point: It demonstrates that it is possible to represent explicit relationships between seemingly unrelated concepts, by abstracting the interpretation away from the concrete ideas.

Furthermore, there are in theory no limits to the kind of relationships that are possible to define in an ontology – anything from *likes*-relations present in social media, to symmetrical *married-to*-relations, is possible. However, in the news domain, linguistic relations are probably the most interesting ones when it comes to extracting semantic knowledge [3].

2.2 REPRESENTATION

The features of a document can be represented in numerous ways. This section covers three possible approaches. First off is the standard *bag-of-words,* followed by its importance-weighting extension TF-IDF. The last approach is a more sophisticated method based on *word embeddings.*

2.2.1 BAG OF WORDS

Bag-of-words (BoW) is a simplification scheme used to represent written documents in a unifying manner [10]. The actual model is rather trivial and consists of creating a *single* vector containing every distinct word across a set of documents. This is then followed by creating a new vector for every document, with corresponding counts for every occurrence of every word contained within.

For instance, consider the two following documents:

(i) *Mary has a little lamb. Mary is five.*

(ii) *Little John's favorite food is lamb chops. Mary's favorite food is pancakes.*

From this create the following vector of unique terms:

$$words = [mary, has, a, little, lamb, is, five, john, favorite, food, chops, pancakes]$$

The corresponding vectors for each of the documents would then become:

	mary	has	a	little	lamb	is	five	john	favorite	food	chops	pancakes
S_1 =	[2	1	1	1	1	1	1	0	0	0	0	0]
S_2 =	[1	0	0	1	1	1	0	1	2	2	1	1]

The final processing step is a normalization procedure, achieved by dividing every value by the length of the respective document. The magnitude of each value in the resulting vector can then be viewed as a measure of the terms' importance within the document.

Although BoW for many years has been the classic way to represent documents, two potential drawbacks remain [3]: First, it treats every word as an independent feature, thus completely ignoring the semantic and syntactic information present in word order and multi-word phrases. Second, the model is oblivious to concepts like synonymy and grammatical nuances. A consequence of these disadvantages is that classification algorithms are limited to inferring patterns in the terminology, as opposed to exploiting the structural semantic knowledge, when extracting information.

2.2.2 TERM FREQUENCY, INVERSE DOCUMENT FREQUENCY

Term frequency, inverse document frequency (TF-IDF) is a popular term weighting metric for feature selection, that can be used in conjunction with the BoW document representation [5] [11] [12] [13] [14]. The metric is built on the assumption that the importance of a term within a corpus is proportional to the number of occurrences in a document, but inversely proportional to the number of documents containing the given term. With other words, a terms measure of importance within a document is dependent on its frequency within the document, as well as the degree of absence in the entire corpus.

The main motive behind using a term weighting metric like TF-IDF is to make it possible to rank and evaluate how important a word is in a document [12]. This is especially useful when classifying text, as words occurring in many documents (e.g. *the, and* or *he*) likely carries less information than those with lower frequency (e.g. name of a sports team, or celebrity). However, it is worth mentioning that TF-IDF has been subject to criticism because of its inability to measure *order* – i.e. the term's position within the text [4]. Yet, its simple nature and long history of use still make it a useful metric [5].

The actual TF-IDF formula comes in many flavors, with varying degrees of complexity [11]. However, the one commonly used in text classification is quite simple, and can be defined as follows: Given a set of terms $t \in \mathcal{T}$ and documents $d \in \mathcal{D}$, where $count(t, d)$ denotes number of occurrences t in d, and vice versa, TF-IDF becomes:

$$TF\text{-}IDF(t_i, d_j) = \frac{\#\ t_i\ in\ d_j}{\#\ t\ in\ d_j} * \log\left(\frac{\#\ documents}{\#\ documents\ with\ t_i}\right) = \frac{count(t_i, d_j)}{\sum count(t, d_j)} * \log\left(\frac{|\mathcal{D}|}{count(d, t_i)}\right)$$

As seen from the formula, the term frequency within a document is normalized over the total number of terms in that document. This normalization step is similar to the one used with the barebones BoW [2.2.1], and results in a metric independent of document length. This is crucial when working with news articles, which can range from small notices, to long, possibly equally important feature articles.

2.2.3 WORD EMBEDDINGS

One of the major shortcomings of traditional feature representations like BoW, is that it purges much of the semantic information present in documents – it does not care whether one word appeared before the other, or the contents of the surrounding sentence.

Word embeddings is a family feature learning techniques that aims to solve this issue [15]. Instead of creating a single feature vector for every document, word embeddings extend the representation by creating a parameterized mapping $W \in words \rightarrow \mathbb{R}^n$ for every *word* – i.e. any higher-dimensional vector.

For instance, given a vocabulary:

$$doc = [john, likes, pancakes, and, hamburger]$$

We might have the following embeddings in $\mathbb{R}^3$:

$$\begin{aligned} john &= [-0.1, 0.3, 0.2] \\ likes &= [0.2, -0.1, 0.6] \\ pancakes &= [-0.7, 0.5, 0.4] \\ and &= [-0.7, 0.5, 0.4] \\ hamburger &= [-0.7, 0.6, 0.3] \end{aligned}$$

It is evident that the embeddings contain *more* information, but it also gives rise to a compelling question:

> *From where do these embeddings originate, and what do they represent?*

The simple answer to the first question is that they are *learned.* Exactly how these embeddings are learned, is in this thesis, subject to the realm of *artificial neural networks*, which is explained in debt in Section 2.5. However, at its core, the matter of calculating the vectors boils down to two different approaches: Either by using *Continuous Bag of Word* (CBOW), which predicts single words from context (i.e. surrounding words), or *Skip-gram,* which predicts the context based on specific words. While both models are capable of producing desired predictions, skip-gram has proven more accurate on bigger datasets [16]. Nevertheless, the end result is a set of embeddings in a high dimensional space.

As for the second question, the intuitive understanding is that the representation should reflect *similarity.* While classic BoW is limited to reflect binary similarity based on concrete words (e.g. whether a document contains "pancakes" or not), the embedded approach is capable of deducing the similarity at a higher level of abstraction. The key idea is that words with similar *meaning* get similar vectors – often measured in terms of either *Euclidean distance* or *Cosine similarity* [17].

For instance, the embeddings may be used to measure the similarity of "man" and "woman", or "king" and "queen" [18]. These measures can then be used to infer the similarity of the sentences "Mathilde of Belgium" and "Sonja of Norway" – both of which in abstract terms means "*queen* of *country*". Subsequently, the same approach can also be used to construct statements previously unknown to the model: For instance, deducing that $king - man + woman = queen$ or even subtle associations like $human - animal = ethics$ and $president - power = prime\ minister$.

Figure 2-2 illustrates this concept by showing how the vectors between the *king* and *queen*, and *man* and *woman* are more or less the same.

As an analogy, this way of thinking is very closely related to how humans process information: If you spot a furry four-legged animal with massive antlers on vacation – that you have never seen before – it is safe to assume that it is a *mammal.* By closer examination, it is probably also possible to infer whether it is closer to a goat than say a

cat. All of this is possible because you, as a sentient being, are capable of combining and incorporating knowledge from past events.

Being able to transfer these principles to the domain of natural language processing, is certainly an intriguing thought.

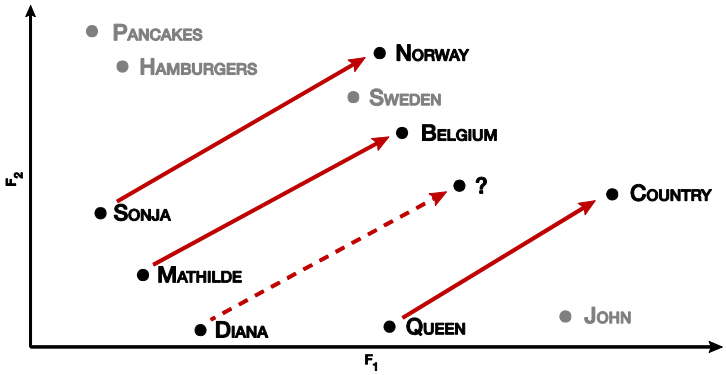


FIGURE 2-2 – CONCEPTUAL WORD EMBEDDINGS RELATIONSHIP, SHOWING SIMILARITY IN 2 DIMENSIONS

2.3 PREPROCESSING

There are as many ways to preprocess a document as there are acorns in a pine forest [19]. However, most of them share a common goal*: Extracting the most substantial features from a document, and outputting a representation suitable for further processing* – commonly known as *feature selection* [20]. This process may range from removing "unnecessary" words, to the task of unifying specific terms with their corresponding synonyms. Throughout this section, a handful of preprocessing concepts and strategies are presented and evaluated, with the underlying purpose of linking their utility to the news article domain.

2.3.1 GROUPING OF SIMILAR TERMS

Every language is bound to have several words and phrases with interchangeable meaning. This can be everything from concepts like synonyms to grammatical nuances.

There are several ways to attack this problem, one being *Stemming* [12]. Stemming is the act of reducing an inflected word down to its root – for instance by using *Porters Algorithm* [21]. In short, Porters Algorithm proposes a technique that works great if the words are *syntactically* similar, like *democracy, democratization* and *democratize.* This leads to complications when faced with interchangeable phrases with significant differences in terms of syntax, like *car* and *automobile.* Thus, the requirement for different tools is made. An increasingly common approach to this is to utilize publicly available ontologies to include more *general* or otherwise common and/or related concepts [3] [12].

Although, subject to controversy [22] [5], the task of grouping terms remains a prominent metric, primarily because of its inherent properties of reducing complexity and feature dimensionality.

2.3.2 STOP-WORD REMOVAL AND TERM FREQUENCY FILTERING

Stop-word[6] removal is the task of filtering out specific words that carry little information in a document [19]. This operation often revolves around prohibiting very common terms, which may range from the exclusion of specific words, to eliminating entire groups of words like prepositions (e.g. *on, in*) and articles (e.g. *the, a, an*). Although a simple procedure in itself, stop-word removal has been proven to be of significant importance in the field of text categorization [23].

It is worth mentioning that there have been several initiatives at work to propose a common list of stop-words [24] [25]. However, a single official list does not yet exist. Furthermore, the use precompiled lists, has even in some cases shown to *reduce* performance on smaller entries of texts, like Tweets [26].

Still, the motivation behind stop-word removal remains high, primarily for its capabilities of reducing the search space, but also to avoid overfitting [19].

Another measure related to stop-word removal is to prune words based on their frequency of occurrence within the corpus [19]. For instance, a word that only appears in a single document is unlikely to yield much, or any information about the other documents. Similar conclusions can be drawn from words occurring in *every* document. Even though term-weighting schemes like TF-IDF [2.2.2] somewhat account for this, they are still unable to actually *reduce* the set of features in the search space. For this reason, performing some kind of frequency based removal as a preprocessing step will likely lead to substantial performance gains in terms of computational demands.

2.3.3 PART OF SPEECH TAGGING

Part-of-speech (POS) tagging is a word sense disambiguation technique that attempts to categorize and tag the words in a document or corpus. For instance, marking word classes such as nouns, verbs, and adjectives.

Primarily, there are two ways to perform the POS-tagging: Rule-based and stochastic. While rule-based taggers dominated the field early on, the handcrafted taggers often end up both high in cost, and in consumption of manpower. One of the reasons for this is

[6] *Stop-words – https://en.wikipedia.org/wiki/Stop_words*

rooted in the many ambiguities present in language [2.1], making rule-based taggers difficult to develop and maintain. This has given rise to an alternative, and cheaper, stochastic approach, sometimes showing comparable performance [27].

Out of the statistical methods, many of the methods commonly used in practice are varieties of the *maximum entropy* tagger developed for the well-known Standford NLP toolkit[7]. Although the choice of POS tags depends on your needs – generally, bigger datasets allow for more fine-grained tagging schemes and vice versa – much of the field have settled with the Brown/Penn tag set [28] shown in Table 2-1.

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

TABLE 2-1 – BROWN/PENN POS TAGS

[7] *Standford NLP – http://nlp.stanford.edu/software/tagger.shtml*

2.4 TRADITIONAL CLASSIFICATION

With the classification of news articles in mind, it becomes apparent that simple single-label techniques will not suffice [29]. Much because an article may have *any* number of correct categories, which in turn perhaps both differ in importance, and inherit correlational properties [30]. For instance, if a news article is labeled *golf,* it is arguably more likely to also have the label *sport*, but less likely to be labeled *politics.*

The diverse and complex composition of this problem admittedly results in a massive forest of possible approaches [30]. However, recent research shows that given enough training data, significant performance can be achieved by even the more primitive techniques [31]. For this reason, the following section covers a few of the simpler, traditional methods, while the succeeding sections are dedicated to shedding light from a state-of-the-art perspective, revolving around neural networks and deep learning.

2.4.1 NAÏVE BAYES

Naïve Bayes, sometimes characterized as the *favorite punching bag of classification techniques,* is one the major cornerstones in the history of statistics and classification [32]. The simple, yet efficient nature of Naïve Bayes – both in terms of classification quality and computational requirements – has been proven successful through a variety of contexts. These range from spam filtering [33] [34] and information retrieval [32], to sentiment analysis [35], with the general application spanning more than forty years of widespread use.

In layman terms, the classification process can be described in three simple steps: (i) Take the probability of an attribute having a given value, for a given classification, proceeded by (ii) multiplying all of them together, followed by finally (iii) multiplying it all with the probability of that classification appearing in the first place.

In more formal terms, for a set of classifications v and a set of attributes a, the Naïve Bayes classifier is merely a simplification of the *Traditional Bayesian* classifier [Figure 2-3]:

$$v_{TB} = argmax_{v_j \in V} P(v_j | a_1, a_2 \dots a_n) = argmax_{v_j \in V} P(a_1, a_2 \dots a_n | v_j) * P(v_j)$$

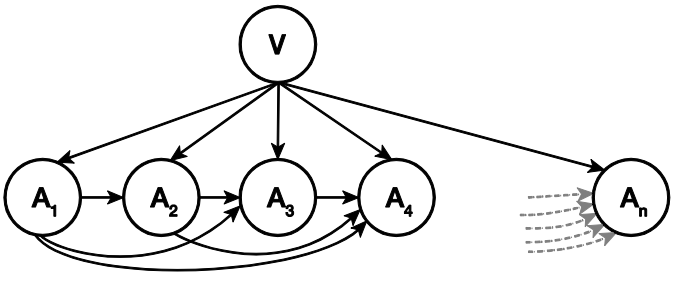


FIGURE 2-3 – TRADITIONAL BAYESIAN WAY TO CLASSIFY WITH CONDITIONAL DEPENDENCIES

Disregarding the conditional dependency, we can simplify and get the following relationship, representing the *Naïve Bayesian* classifier [Figure 2-4]:

$$v_{NB} = argmax_{v_j \in V} P(v_j) \prod_i P(a_i|v_j)$$

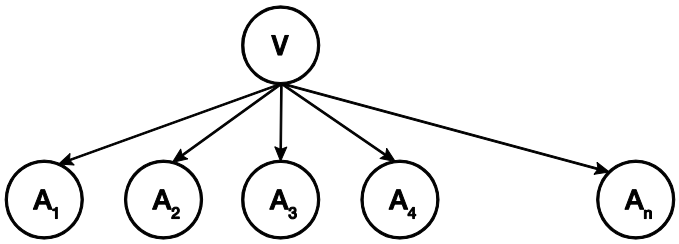


FIGURE 2-4 – NAIVE BAYES APPROACH TO CLASSIFYING, WITHOUT THE CONDITIONAL DEPENDENCIES

The assumption of conditional independence might sound like a longshot – certainly, some of the attributes in any given domain are bound to be related. However, the simplicity of the Naïve Bayes still holds ground, and is shown to be a very successful classifier, regardless of internal interdependencies [32] [36].

There is a catch though: The classifier in its original form only handles single categories – that is, it only assigns single categories to a document, not multiple. However, research shows that by transforming the problem into n-binary classification problems – one for each target category – the Naïve Bayes classifier becomes very capable of handling multi-label classification task as well [34].

As for the configuration of the classifier, there are two main options: *multinomial* and *Bernoulli* [37]. The multinomial version supports the use of feature vectors with decimal weights, and produces a binary classification output. On the contrary, the Bernoulli model only allows for binary feature vectors, but produces a probabilistic classification output. The sensible choice when working with text data represented in a BoW fashion, is a multinomial configuration, because of incorporation of frequency information (i.e. decimal weights) [37].

2.4.2 SUPPORT VECTOR MACHINES

Support Vector Machines (SVM) originates from the family of *large margin classifiers* and are used to find linear decision surfaces (i.e. hyperplane) with the highest margin between the positive and negative classification instances [Figure 2-5] [30]. To achieve this, an SVM exploits the idea of *kernel functions,* which are used to manipulate and project data points into another space. This procedure results in the SVM becoming linearly separable by a single vector [Figure 2-6].

SVMs are especially suited for text categorization, mainly for their capabilities of tackling classification problems independently of the dimensionality of the feature space [38]. This makes them a very robust and flexible tool in the linguistic domain, considering the potential sparseness and dimensionality of document vectors [39].

Moreover, the trait of capturing subtle and possibly rare details in a document has been shown to be a lot more crucial than previously thought – in particular on the web, where much of the data consist of individually rare, but collectively frequent events [31].

Similar to the Naïve Bayes classifier, SVMs are initially suited for single-label classification. However, multi-label modifications have been proposed in the literature, which tends to involve either transformation to n-binary problems, to through kernel-modifications [29].

Although, an excellent classifier, SVMs are generally slower than simpler methods like Naïve Bayes, which leads to its practical use being somewhat constrained by the size of the corpus [29].

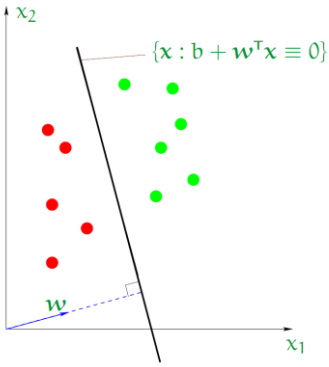


FIGURE 2-5 – SVM IN TWO-DIMENSIONAL SPACE WITH A LINEAR KERNEL FUNCTION

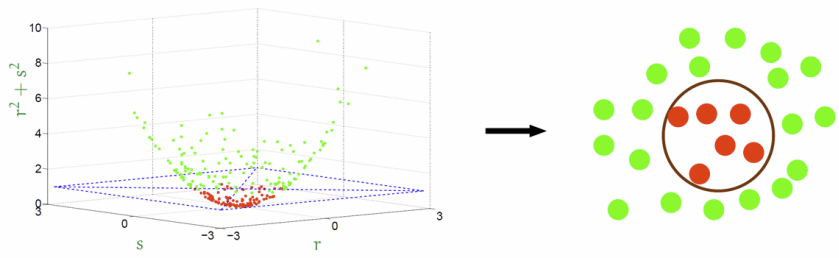


FIGURE 2-6 – SVM CLASSIFICATION EXAMPLE FROM THREE-DIMENSIONAL SPACE

2.4.3 K-NEAREST NEIGHBOR

K-Nearest Neighbor (k-NN) is lazy learning method, which means that it does not build any explicit representation of categories during the training phase [40]. Since there is no actual model, keeping the classifier up-to-date is trivial: Once the algorithm has classified a new unseen instance, the "training" is simply a manner of indexing the example in the database.

The classification mechanisms in k-NN rely on exploiting the *closeness* of k near neighbors given a certain feature space. For instance, by picking the k closest entities based on the Euclidean distance. Once k near neighbors are chosen, the task of classifying is commonly accomplished by having a majority election [40]. Another possible approach is distance weighting, where nearby neighbors are more worth than those far away [41].

One of the biggest strengths of the k-NN classifier is that it makes very few assumptions about the actual data [5], as opposed to for example Naïve Bayes, which assumes independence between attributes. In fact, the only requirement k-NN has in terms of data, is that a reasonable similarity measure exists.

However, its simplicity is also its biggest weakness. Since there is no preprocessed model, the classification of a new instance is by default dependent on every other example in the training set. With a big enough database, the computational deficiency at testing time, can quickly become a huge bottleneck. Some research does, however, show that this can be partially solved by taking advantage of clustering as a preprocessing step [42].

Nevertheless, due to its popularity and long history of use [13], it is still worth investigating, as it provides a good baseline when comparing it with other classifiers. Additionally, initiatives have been made to modify the algorithm for a multi-label environment [43].

2.5 ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANNs) is a family of machine learning algorithms inspired by the biological network present in our brain. Conceptually, an ANN is a directed network of interconnected nodes, known as *neurons* – with every neuron acting as an independent computational unit. Transparent as this may seem, history shows that by combining these neurons in certain ways, a structure inheriting properties akin to those of a mathematical function emerges: A deterministic structure capable of producing some output given some input – much like the neurons and synapses in our brain interprets the world through our senses.

Although the principles of neural computing emerged already in the 1940s [44], the initial breakthrough was Rosenblatt's introduction of the *perceptron* in 1958 [45]. The perceptron was a simple component, which turned its inputs into a single binary output, based on whether or not the weighted sum of the inputs was above some given threshold. Researchers quickly began experimenting with these neurons, believing it was the first

step towards being able to model the human brain. Hence, the years following this invention led to a surge in hype and enthusiasm throughout the field of artificial neural research.

However, during the mid-1960s the excitement was suddenly brought to a halt – mainly due to the binary perceptrons' inability to model even basic concepts like the XOR-gate [44]. In addition, these early networks were incredibly difficult to *train*. From there on the field remained more or less desolate until the mid-1980s, with the discovery of *backpropagation* [46], and the fact that researchers finally managed to prove that a multilayer feedforward networks were universal approximators [47].

Despite the surge in potential, ANNs have mostly remained difficult to work with due to their complexity and demanding requirements regarding training data and computability. That is, until recently. With much help from Moore's law[8] and major advancements in data processing efficiency, ANNs have finally become a tractable option for real life applications. As a result, ANNs have over the past few years, shown to outperformed *state-of-the-art* techniques across a wide variety of fields – ranging from speech recognition and object detection to drug discovery and genomics [48].

Needless to say, the capabilities and potential of ANNs span wide and far, with the field growing more complex every day. For this reason, a thorough elaboration of the current state-of-the-art in neural computing is required to fully comprehend the motivation behind its application to the news domain. Thus, the following sections attempt to do exactly that.

2.5.1 GENERAL OVERVIEW

PERCEPTRON

To recap, the perceptron is the simplest kind of neuron, and is the primary building block in neural nets. For the perceptron each of the $n \in \mathbb{N}$ inputs are scaled with a weight $w_n \in \mathbb{R}$ according to their *importance,* followed by computing the weighted sum $\sum_j w_j x_j$ [Figure 2-7].

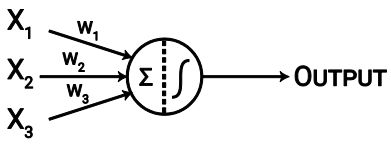


FIGURE 2-7 – SIMPLE ANN WITH A SINGLE NEURON, WITH 3 INPUTS

[8] *Moore's law – https://en.wikipedia.org/wiki/Moore%27s_law*

The sum is then run through a *threshold* function within the neuron – for instance outputting a 1 if the sum is above some given threshold, or 0 otherwise. In algebraic terms:

$$output = \begin{cases} 0 & if \sum_j w_j x_j \leq threshold \\ 1 & if \sum_j w_j x_j > threshold \end{cases}$$

The notation can be simplified and rewritten in terms of vectors and biases by defining the following relationships:

$$\boldsymbol{w} * \boldsymbol{x} = \sum_j w_j x_j$$
$$\boldsymbol{b} = -threshold$$

The simplified representation of a perceptron would then become:

$$output = \begin{cases} 0 & if\ \boldsymbol{w} * \boldsymbol{x} + \boldsymbol{b} \leq 0 \\ 1 & if\ \boldsymbol{w} * \boldsymbol{x} + \boldsymbol{b} > 0 \end{cases}$$

The function computing the output is known as the *activation function.* Although the perceptron by definition produces a binary output, other types of neurons can, in theory, inherit *any* activation function [2.5.2].

NETWORK

Using the principles of the perceptron, the same idea can be extended and generalized to an arbitrary number of neurons, inputs, outputs and activation functions, ultimately creating a network. Figure 2-8 shows an example of a 3-layered network where all the nodes in layer n are connected to every node in layer $n + 1$ – formally known as a *bipartite graph.* If a non-linear activation function is used, the resulting network is a *multilayer perceptron,* or *feedforward network* (FFN). The layer in between the input and output layers is called a hidden layer.

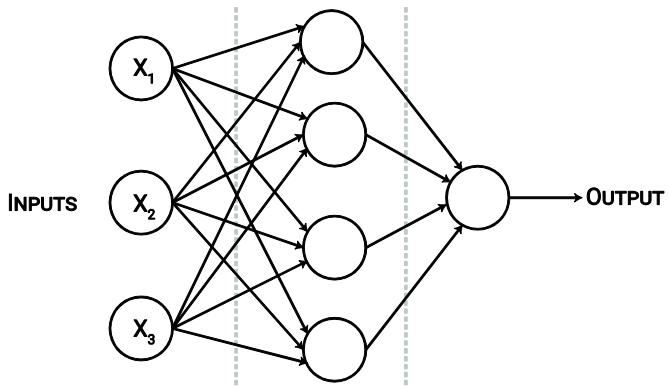


FIGURE 2-8 – 3-LAYER NEURAL NETWORK, OR A MULTILAYER PERCEPTRON

The motivation behind the layered structure is based on the notion that each and every layer represents a *level of abstraction*. For instance, the first layer may understand how to find lines and edges in a photo, while the next layer combines the lines together forming geometric objects. A third layer may then use information to decide whether or not the final image is something like a digit or a cat.

Although the above example is comprehensible and easy to understand, the real strength of ANNs surfaces when creating a network with many layers: A *deep* neural network. Utilizing such a network with two or more hidden layers is what the literature today refer to as *deep learning* [48]. An example of a deep neural network is depicted in Figure 2-9.

By feeding such a deep network with massive amounts of data, the network might be able to find relations previously unknown to humans – all in an automated fashion. However, to get an intuition of how this is possible, a closer look into the activation functions and training algorithms is required.

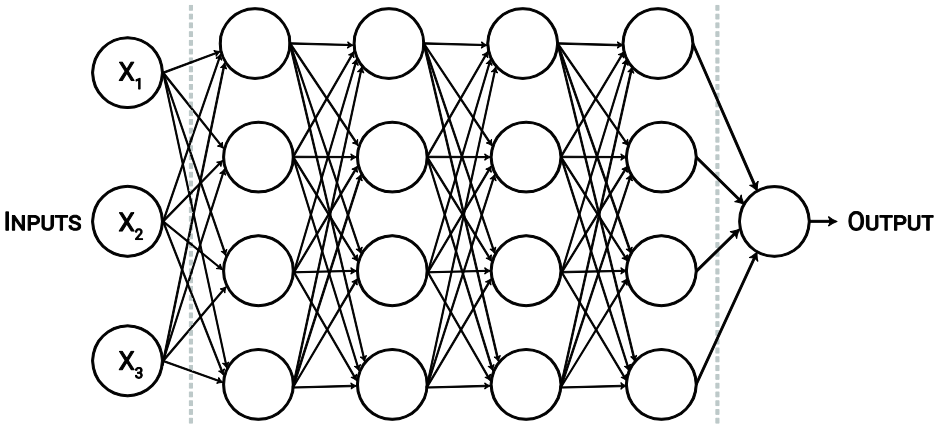


FIGURE 2-9 – DEEP NEURAL NETWORK WITH 4 HIDDEN LAYERS

2.5.2 ACTIVATION FUNCTIONS

The purpose of an activation function is to transform the activation level of a neuron into an output signal. While there are currently numerous possible activation functions to choose from, neural practitioners tend to stick with only a handful of them [49].

With the binary activation function used in perceptrons, a tiny change (i.e. output changing from 0 to 1) could cause major ripple effects throughout to the whole network, making learning almost impossible [50]. A more desirable function to better facilitate learning would be a one capable of outputting *any* real number between 0 and 1. As of now, the functions proven most useful are the S*igmoid, hyperbolic tangent, Softsign* and *rectified linear units* (ReLU).

SIGMOID

The Sigmoid, or sometimes called the *logistic* function has by far been the most common activation function [49]. It is a non-linear function with exponential properties, and is defined by:

$$\sigma_s(z) = \frac{1}{1 + e^{-z}} \tag{2.1}$$

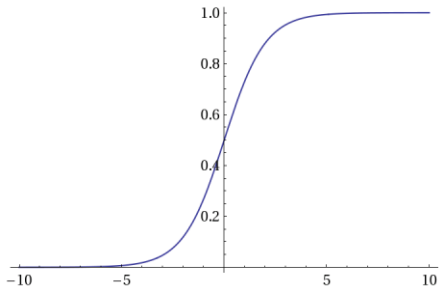


FIGURE 2-10 – SIGMOID FUNCTION

HYPERBOLIC TANGENT

The hyperbolic tangent is in principle the same as the Sigmoid, with the exception that it is defined from -1 to 1, instead of 0 to 1:

$$\sigma_t(z) = tanh(z) \tag{2.2}$$

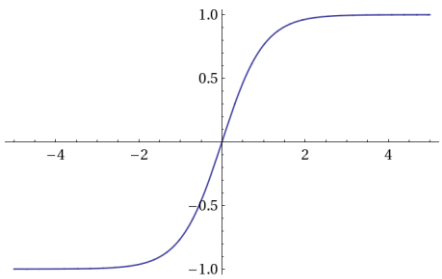


FIGURE 2-11 – HYPERBOLIC TANGENT ACTIVATION FUNCTION

The motivation behind the hyperbolic tangent is to avoid *systematic bias,* with some research showing that using a symmetric activation function could solve this and lead to faster convergence [51].

SOFTSIGN

A close relative to the hyperbolic tangent is the *Softsign* function – with the only difference being that its tails are quadratic polynomials rather than exponential [52]. This means that the Softsign approaches its asymptotes much slower than the hyperbolic tangent. Formally it is defined as:

$$\sigma_{ss} = \frac{x}{1 + |x|}$$

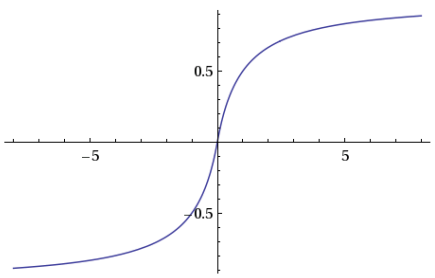


FIGURE 2-12 – SOFTSIGN ACTIVATION FUNCTION

RECTIFIED LINEAR UNITS – RELU

Although the Sigmoid has been a prominent activation function for a long time, the default recommendation for modern neural networks is often to use the *Rectified Linear Units* (ReLU) [53]:

$$\sigma_{relu}(z) = max\{0, z\} \tag{2.3}$$

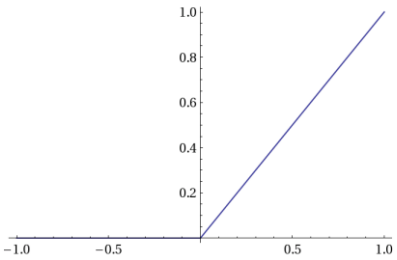


FIGURE 2-13 – RELU ACTIVATION FUNCTION

As shown in Figure 2-13, the ReLU consist of a piecewise linear function with two linear components. For this reason, ReLU preserves many of the properties that make linear models easy to optimize with gradient descent, as well as generalizing well. Another important advantage is that ReLUs are computationally inexpensive in comparison to its continuous counterparts, and thus suitable for bigger networks [54]. The computational convenience is further amplified for naturally sparse datasets, as the ReLU cuts off at exactly zero – making it possible to use a sparse encoding of the internal representation.

2.5.3 OUTPUT FUNCTIONS

Output functions are in principle identical to activation functions, with the only difference being that they compute – as the name suggests – the *output* of the entire network. For instance: Given a multi-label classification model, one might want to output a probability distribution over all of the possible labels, as opposed to a number between 1 and 0 – like a Sigmoid.

For this reason, it has been common practice to use the *softmax* function – also known as *normalized exponential* – on the output layer [55]. Succinctly, the softmax function is a generalization of the Sigmoid that works for multiple output classes. While the Sigmoid model outputs a single value (e.g. 0 or 1 for a single class), the softmax outputs a probability distribution over any set of possible classes. For a set of K classes (i.e. outputs in the networks), softmax can be described as *squashing* a K-dimensional vector $\boldsymbol{z}$ of real values into a K-dimensional vector of $\sigma(\boldsymbol{z}) \in (0,1)$, where $\sum_i \sigma(z_i) = 1$. In general, the softmax function is given by:

$$\sigma_{sm}(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j \in \{1, \dots, K\}$$

2.5.4 USING AN ARTIFICIAL NEURAL NETWORK

Using an already configured and trained ANN is in theory as simple as providing some vectorized input, followed by observing the output on the other end. For instance, by inserting an image and then make the network decide whether or not the image contains a cat [56]. However, to make the network produce a sensible output, four complicated matters need to be resolved:

(i) *How to vectorize the input?*

(ii) *What should the output be?*

(iii) *What should the network look like?*

(iv) *What should the weights be?*

The answer to the first two questions depends on the chosen media. For an image, one could just create a single vector from the pixel matrix, and for text, one could insert the feature vectors produced by BoW and TF-IDF. The output could be binary, as in: "is it a cat or not?". Although not necessarily trivial, the process itself is straight forward.

As for the third and fourth question: We simply do not *know*. There are no set of "optimal weights" or an "optimal configuration" – everything depends on the properties of the underlying data, and what one seeks to achieve from it. Though, while the configuration of the network is a somewhat an experimental "trial-and-error" process, the weights need to be *learned*.

Luckily, there are several guidelines and best practices as to how one could tackle both of these problems, and the following sections will shed some light on the most important ones.

2.5.5 TRAINING

Over the years, there have been numerous more or less exotic ways of training ANNs. The approaches have ranged from semi-chaotic evolutionary techniques [57] to the classical, more controlled combination of *gradient descent* and *backpropagation* [58]. However, most of the field has settled with the latter, both due to its efficiency and properties that promote distributed training [50].

Apart from that, training an ANN is analogous to training any other modeling algorithm: Start by feeding it training examples, and measure the error of the output with a *cost function*[9]. As with the activation function, there are numerous possible cost functions. However, for classification-related ANN tasks, research points in favor of the *cross-entropy error* (CEE), or the classic *mean square error* (MSE) – with the former having the upper hand. This is mainly due to CEEs property of promoting rapid learning, even when the network is unambiguously wrong – a property not inherited by the MSE [50]. With other words: If the values of output function are a lot different than the correct output, the CEE makes it possible for the learning algorithm to take a bigger "step" in the right direction than the MSE generally is capable of.

For a single neuron with m training examples, where y denotes the *real* output and a is the approximated output generated from the network, the CEE is defined as [56]:

$$CEE = -\frac{1}{m}\sum_{k=1}^{m}[y_x \; ln(a_x) + (1 - y_x) \; ln(1 - a_x)] \tag{2.4}$$

The reasoning behind why CEE works is described thoroughly in [50]. However, the general takeaway is that the partial derivatives of CEE generally remain greater in magnitude than MSE when the error is huge. As shown in the next section, this property becomes useful when training the network with gradient descent, where the partial derivatives of the cost function play an integral role.

LEARNING WITH GRADIENT DESCENT

To actually train an ANN, the cost function has to be minimized over all the neurons, that is, all weights and biases for the entire network. The preferred method for doing so is *gradient descent* [50]. The easiest way to understand how and why gradient descent

[9] *Also known as loss or objective function*

works is with an example: Consider a vector $\boldsymbol{v} = (v_1, v_2)^T$ defining the feature space illustrated in Figure 2-14.

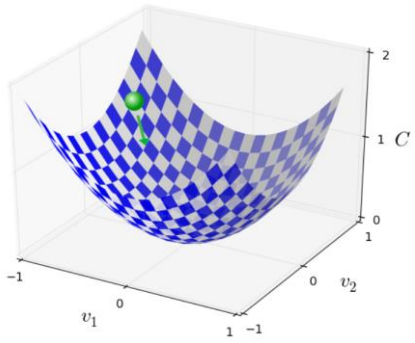


FIGURE 2-14 – FEATURE SPACE FOR $\boldsymbol{v}$

The idea behind gradient descent to compute the *gradient*, i.e. the direction of the "steepest" slope for the current position, and then move a tiny step in the *opposite* direction.

For vector $\boldsymbol{v}$ and cost function $\boldsymbol{C}$ that would translate into first defining the change in cost $\Delta\boldsymbol{C}$:

$$\Delta\boldsymbol{C} = \frac{\delta C}{\delta v_1}\Delta v_1 + \frac{\delta C}{\delta v_2}\Delta v_2$$

Followed by defining the *gradient* of $\boldsymbol{C}$, with T being the *transpose* operator:

$$\nabla\boldsymbol{C} = \left(\frac{\delta C}{\delta v_1}, \frac{\delta C}{\delta v_2}\right)^T$$

And letting $\Delta\boldsymbol{v}$ be the change in $\boldsymbol{v}$:

$$\Delta\boldsymbol{v} = (\Delta v_1, \Delta v_2)^T$$

Using these relationships, $\Delta\boldsymbol{C}$ can be rewritten as:

$$\Delta\boldsymbol{C} = \nabla\boldsymbol{C} * \Delta\boldsymbol{v} \tag{2.5}$$

As mentioned, the goal is to move in the *opposite* direction of the gradient – with other words, a *negative* $\Delta\boldsymbol{C}$. This can be done by defining the following relationship, where η is a small, positive parameter known as the *learning rate*:

$$\Delta\boldsymbol{v} = -\eta\nabla\boldsymbol{C}$$

Finally, the above equation can be used to create what is known as the *gradient update rule* – the quantity to update $\boldsymbol{v}$ in order to take one tiny step *downhill*, thus minimizing the cost. By letting $\boldsymbol{i}$ denote the current iteration, the update rule becomes:

$$\boldsymbol{v}_{i+1} = \boldsymbol{v}_i - \eta\nabla\boldsymbol{C} \tag{2.6}$$

By repeatedly applying the update rule on a convex error function, like MSE or CEE, the error has been shown always to converge, given that the learning rate is low enough [50]. However, the number of iterations required for this convergence is subject to trial and error, as it depends on the particular problem at hand. For this reason, it is often helpful to plot the error function while training, giving a graphical perspective of the movement of the gradient.

Although the derivation above only used a two-component vector, gradient descent can be generalized to an arbitrary number of components. For the ANN case, the vector would consist of all of the dynamic features in the network, namely, all weights and biases. Adopting the same procedure as above, this results in the following two update rules:

$$w_{n+1} = w_n - \eta \frac{\delta \boldsymbol{C}}{\delta w_n}$$
$$b_{l+1} = b_l - \eta \frac{\delta \boldsymbol{C}}{\delta b_l} \tag{2.7}$$

OVERCOMING CHALLENGES WITH STOCHASTIC GRADIENT DESCENT

Even though gradient descent in its pure form, as described above, is going to minimize the error rather efficiently, there is one major drawback. The cost function is directly dependent on *every* training instance, which means that the gradients need to be computed separately for every training input, before being averaged. This also means it will be difficult to run the training in a distributed environment.

The solution for this is to split up the training instances in randomly chosen disjoint *batches,* or *mini-batches,* followed by running gradient descent on each of the batches. An *epoch* denotes a single run through the entire dataset. By averaging over all of the gradients after each epoch, gradient descent can be executed in a distributed manner – which often results in both *faster* and even *better* learning [59] [60]. This method of applying gradient descent is known as *stochastic gradient descent* (SGD) – *stochastic* because each small set of examples gives a noisy estimate of the average gradient over all examples [48]. In more formal terms: For a mini-batch with size $\boldsymbol{m}$, the gradient (2.5) for SGD is then redefined as:

$$\nabla \boldsymbol{C} = \frac{1}{m} \sum_{j=1}^{m} \Delta \boldsymbol{C}_{x_j} \tag{2.8}$$

And the update rule (2.7) for SGD becomes:

$$w_{n+1} = w_n - \frac{\eta}{m} \sum_{j=1}^{m} \frac{\delta \boldsymbol{C}_{x_j}}{\delta w_k}$$
$$b_{l+1} = b_l - \frac{\eta}{m} \sum_{j=1}^{m} \frac{\delta \boldsymbol{C}_{x_j}}{\delta b_l} \tag{2.9}$$

COMPUTING GRADIENTS WITH BACKPROPAGATION

Gradient descent provides a framework for error measurement and an intuition of how to update weights and biases using gradients. However, this is not of much use without also knowing how to *compute* the actual gradients of the cost function. One way to calculate the gradients would be by choosing some small error $\epsilon > 0$, and a unit vector in the jth direction e_j, and then use the approximation:

$$\frac{\delta \boldsymbol{C}}{\delta w_j} \approx \frac{\boldsymbol{C}(w + \epsilon e_j) - \boldsymbol{C}(w)}{\epsilon}$$

This approach would *work,* but given n weights it would also require the cost function to be calculated n times for *a single* training example. This would undoubtedly quickly become very ineffective and possibly insoluble for a cost function with many variables.

Backpropagation solves this issue by granting a way to apply the same principle, with a *single* pass through the network [58]. In simple terms, the algorithm does the forward propagation of the error *backward.* This is done using efficient techniques from linear algebra to repeatedly apply the *chain rule* from calculus to compute the partial derivatives required for every gradient.

The actual mathematical derivation behind the algorithm is rather involved, and lies beyond the scope of this thesis [61, 62]. Nevertheless, for the purpose of the thesis, it is sufficient to envision backpropagation as a black box that computes gradients, *efficiently.*

2.5.6 TOPOLOGIES

This far, the only ANN architecture, or *topology* discussed, has been the classic *feedforward* multilayer perceptron (FFN) – a network where all the nodes are connected to every node in the next layer. No loops, and no partially connected layers. However, over the years there has been no shortage of more sophisticated network topologies. For the scope of this thesis, two of the more successful ones are investigated: Namely, *convolutional* and *recurrent* neural networks, with the main focus being on the latter.

CONVOLUTIONAL NEURAL NETWORKS

As a long time popular choice for image classification, *convolutional neural networks* (CNNs) is a type of neural network especially suited for working with multi-dimensional data [54]. For example, images with three channels of 2D arrays denoting color intensities, but also 1D data like text or volumetric content like video.

The idea behind CNNs is to systematically identify and exploit increasingly abstract relationships in data. A classic example for explaining this procedure is image classification: The first layer may detect simple concepts like lines and curves throughout an image. By combining these lines and curves in the second layer, the network is able to identify geometrical shapes like squares and circles, which in the third layer might translate into more complex features like an ear or an eye. Carrying on, the network

might be able to question whether the image contains a cat or a tricycle down the line. This may sound similar to how the layered neural architecture was explained in Section 2.5.1, and it *is* – CNNs just takes this idea to the extreme.

Mathematically speaking, convolution is an operation that expresses the amount of overlap of one function, as it is shifted over another function. The output of the operation is a *similarity* measure between the two functions. In terms of the image classification use case, imagine convolution as sliding a small window (or *filter*) over an image, systematically capturing a subset of the pixels (typically on the order of 5 by 5) at the time. And for every new position, the window analyses the underlying pixels, and answers the question "does this subset contain feature x".

In principle, a CNN is based on convolving multiple such windows asking different questions – forming a *convolutional layer*. The output of the convolutional layer is then passed on to a *pooling layer* which collects all of the "answers" from the convolutional layer, and merges semantically similar signals. The idea behind the merge is that the location of features should not matter – it does not matter whether an eye is in the upper right or lower left, as long as it is above the mouth. By stacking convolutional and pooling layers, a CNN is formed.

It should be noted that the above example is heavily simplified. While CNNs are both widely applicable and fascinating to understand in depth, the topological focus for this thesis has been chosen to be on RNNs, introduced in the next section. Thus, for a more fine-grained dive into the domain of CNNs, the reader should consult the original works of A.I. pioneer Yann LeCun et al. [62].

RECURRENT NEURAL NETWORKS

Recurrent neural networks (RNNs) is a kind of network especially suited for working with sequential data – it be a time series, continuous sound or written language. The main difference from the standard FFN, is that RNNs has *backward edges* – the input depends on both the input, *and* the output from the previous iteration. Due to this, RNNs inherit a kind of *memory*, and thus is capable of making inferences based on the current input *given* its previous output – or rather, the present, given its recent past.

For instance, RNNs can be used to predict the next letter of a word, given the n previous characters in the sentence. Another way to look at it is to say that RNNs preserves contextual information related to the *order* of the input. In contrast to FFNs which simply outputs static information based on current input, a RNN is capable of dynamically adjusting its output based on what it *remembers* – a model closer related to the human mind than traditional approaches.

At first, it may sound courageous to feed the output back into the network, as it would indeed create loops [Figure 2-15]. The key here is to remember that the output is only redirected through the net at the *next* time step. Because of this, is it possible to *unfold* any RNN into a chain of n separate networks, where n denotes the finite number of time steps the network "remembers" [Figure 2-16].

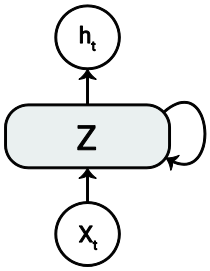


FIGURE 2-15 – SIMPLE RNN WHERE Z IS A NEURAL STRUCTURE, DIRECTING OUTPUT BACK INTO THE NETWORK.

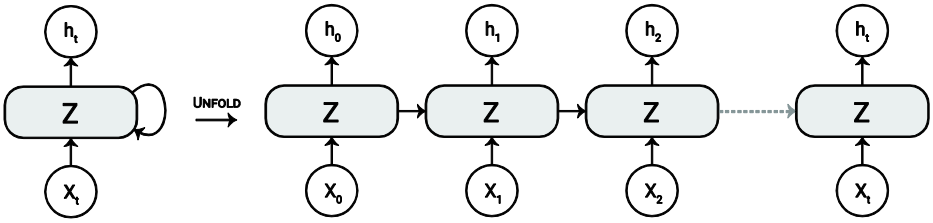


FIGURE 2-16 – UNFOLDED RNN STRUCTURE

Another important property of RNNs is the capability of dynamically dealing with a variable length of input and output vectors [Figure 2-17]. For instance, making it possible to input several values at once, only to produce a single, perhaps nominal output, like a category. Viewed from a developer's perspective, the process can be seen as analogous to how an actual computer program works. In fact, RNNs are known to be *Turing-Complete* in the sense that they can simulate arbitrary programs, with some input, an internal state, and some output [63].

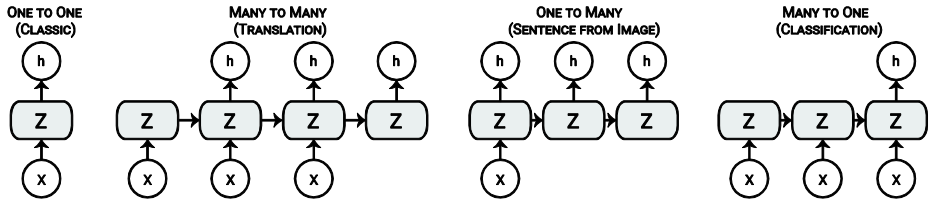


FIGURE 2-17 – POSSIBLE INPUT AND OUTPUT CONFIGURATIONS FOR RNN

As a result, RNNs are especially suitable for classifying text: Either by inputting entire documents, or by using ordered feature vectors, while outputting a single or more categories.

When it comes to training an RNN, the process is more or less identical to how one would train a standard FFN. The only change required is an extension of the backpropagation algorithm, known as *backpropagation through time* (BPTT) [58] – which purpose is to link the different time-steps together.

Although it is easy to become blinded by the benefits of RNNs, there are a few drawbacks to consider. Perhaps the most central one is the difficulty of training. Since the training examples are fed through the network multiple times, the number of multiplications required to compute the gradient also increases. As shown in (2.6), the gradient is calculated as a function of the derivatives of the activation functions and the related weights. If either of these factors is < 1, the gradient may converge towards zero, leading to what is known as the *vanishing gradient problem*. In the opposite scenario, where both factors are > 1, the gradient might increase uncontrollably, resulting in the *exploding gradient problem* [64] [50]. Intuitively, this phenomenon is somehow analogous to why consistently winning 0.97 or 1.03 NOK for every krone spent when gambling quickly either leads to bankruptcy or incredible wealth.

The reason that gradients with extreme values is an issue can be deduced from the update rule in gradient descent (2.7). If the gradient is tiny, the updates of the parameters will also be tiny, and the opposite for very high values. As the whole training process is dependent on the gradient to move towards a solution, small updates will lead to very slow training, while large gradients will lead to big fluctuations and an unstable network.

LONG SHORT-TERM MEMORY UNITS

The issues of vanishing and exploding gradients introduced in the previous section is a fundamental problem with bigger neural networks. Luckily, prevention mechanisms exist: By swapping the traditional neurons in an RNN with *long short-term memory units* (LSTMs) research shows RNNs can be made both resistant to vanishing and exploding gradients, as well as making the RNN easier to train [65] [48].

In short, LSTMs are *smart* structures, capable of controlling their own behavior, and managing their own memory. Specifically, LSTMs are capable of remembering a value of an arbitrary length of time, or choose to forget the value if deemed insignificant.

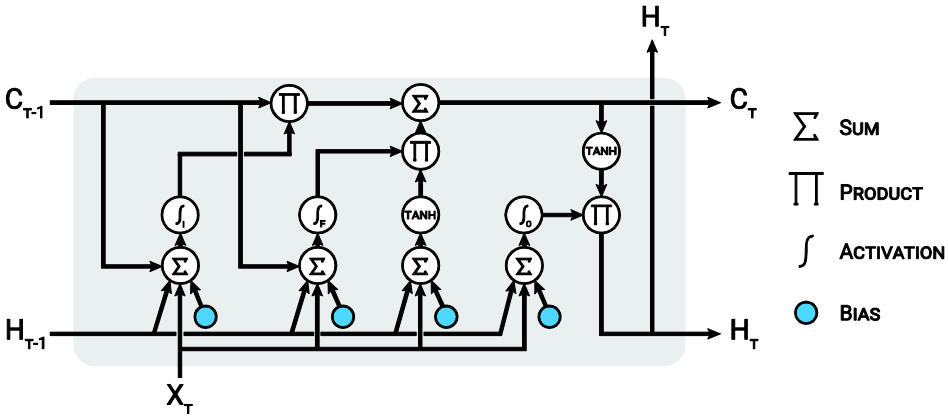


FIGURE 2-18 – THE INNER ARCHITECTURE OF AN LSTM UNIT

As illustrated in Figure 2-18, the design of an LSTM unit is far from trivial at first glance. Perhaps the first thing that comes to mind is the fact that there are three inputs:

$$h_{t-1} = outputLastTimestep$$

$$x_t = outputPreviousLayer$$

$$c_{t-1} = memoryLastTimestep$$

With two corresponding outputs:

$$h_t = outputCurrentTimestep$$

$$c_t = outputMemory$$

Within the unit itself, there are also three internal units known as *gates* (i.e. the sigmoid-shaped units). Due to the properties of a typical activation function (like the Sigmoid, the output is either very close to 1 or 0), a gate is comparable to a binary control unit – merely as a side effect of the component-wise multiplication. That is, if a gate *fires* its output is going to be close to 1, and the multiplicative step inside the next node will be left unaltered. On the contrary, if a gate does not fire, the output is going to be close to 0, which effectively blocks the signal through the next node.

Summarized, the gates control three different properties: The first controls whether or not the unit should care about its current inputs, the second controls the unit's internal memory, while the third controls whether the neuron should fire or not.

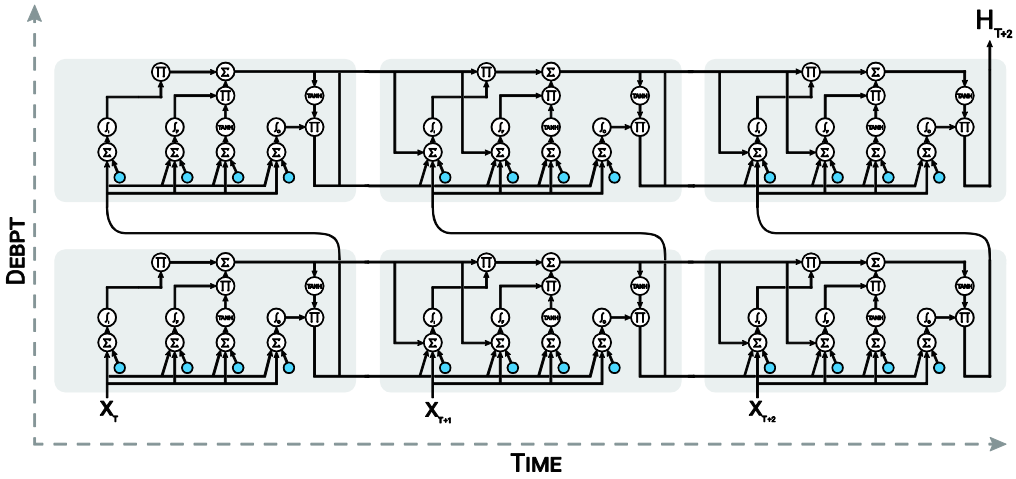


FIGURE 2-19 – LSTM COMBINED, PRODUCING AN RNN WITH 3 INPUTS AND 1 OUTPUTS. E.G. FOR CLASSIFICATION.

The reason LSTMs prevent the vanishing and exploding gradient problem is twofold: First, the output is not directly dependent on the output of the activation function, but rather an identity function which derivative is always 1. Second, the weights of the LSTM are effectively equal to the *memory gate* which value is always ≤ 1. For this reason, the gradient neither vanishes nor explode.

Finally, like any neural structure, LTSMs can be combined in arbitrary ways, forming configurations like the ones in Figure 2-17. An example LSTM configuration with three inputs, and a single output is depicted in Figure 2-19 – a typical topology suitable for classification.

In light of the above, LSTMs becomes the natural choice when working with RNNs today.

2.5.7 WEIGHT INITIALIZATION AND COMMON CHALLENGES

When training a fresh ANN, the network itself is free of knowledge. Still, the weights and biases have to be initialized to some value. It is tempting to conclude that starting from a set of random weights, and letting gradient descent work from there, is a sensible choice. However, as it turns out: Clever initialization of weights may matter a lot in terms of convergence – both in regards to speed, and whether or not the network will converge at all [66]. The following section covers some of the commonly used initialization techniques.

XAVIER

A popular alternative to random initialization is *Xavier initialization* [52]. Xavier initialization is similar to a Gaussian, but *narrower* – i.e. fewer values close to the borders of 0 and 1. Formally: Given a set of weights w and a neuron with n_{in} in-edges and n_{out} out-edges, sample weights from either a Gaussian or uniform distribution, with a zero mean and a specific variance:

$$Var_{xavier}(w) = \frac{2}{n_{in} + n_{out}}$$

In summary: Using a narrower Gaussian approach like Xavier makes every iteration affect the weights in the network more heavily, leading to faster training.

RELU

One problem with the Xavier initialization is that it assumes that the activations (approximately) are linear. This is fine when the $sigmoid$ or $tanh$ is used, but it does not hold for ReLUs. However, all else being equal, the only modification required to make it compatible with ReLU is to change the variance to [67]:

$$Var_{ReLU}(w) = \frac{2}{n_{in}}$$

2.5.8 OPTIMIZATION

Many of the optimizations and guidelines discussed this far has been static parameters that need to be set prior to training. Still, knowing all of the information required to achieve a satisfactory training environment up front, is often undeniably ambitious. For this reason, several more or less dynamic and automatic tuning algorithms have been developed – with a common goal of further adapting the ANN to the particular task at hand.

MOMENTUM

A natural property of gradient descent that it often ends up traversing the feature space in a zigzag-like pattern [50]. This movement is not optimal, as it increases the distance the training algorithm has to travel, which in turn increases the number of steps or iterations required to find a solution. One technique to smoothen out the path, and thus lower the traveling distance during training, is to apply momentum [66].

From physics, we know that momentum is the property that makes the car skid if you quickly rotate the turning wheel – the car wants to continue straight ahead, while the tires want it to shift direction. What you end up with is some sort of mix, where the car is slowly changing its direction, instead of doing an immediate turn. The same idea can be applied to the gradient by incorporating knowledge from its previous step into the calculation of the current [Figure 2-20].

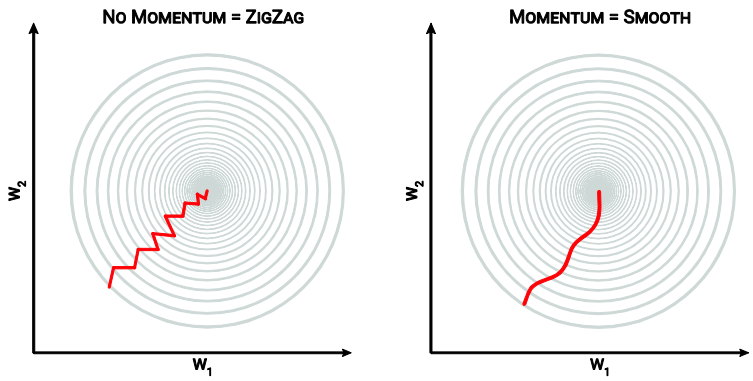


FIGURE 2-20 – THE EFFECTS OF APPLYING MOMENTUM TO THE GRADIENT DESCENT, TO SMOOTHEN THE PATH

The classical momentum is incorporated in training with the following simple modification to the gradient descent update rule (2.7) – where $\mu \in \{0, \dots, 1\}$ is the momentum coefficient:

$$\boldsymbol{m}_{i+1} = \mu \boldsymbol{m}_i + \eta \nabla \boldsymbol{C}$$
$$\boldsymbol{v}_{i+1} = \boldsymbol{v}_i - \boldsymbol{m}_{i+1}$$

ADAPTIVE LEARNING RATE

Once faced with a new problem, the initial choice of learning rate is often a shot in the dark. A learning rate too big is going to overshoot the minimum, while a learning rate too small will lead to very slow training [Figure 2-21] – hence the intuitive solution would be to take big steps in the beginning, and adjust the rate when approaching the minimum.

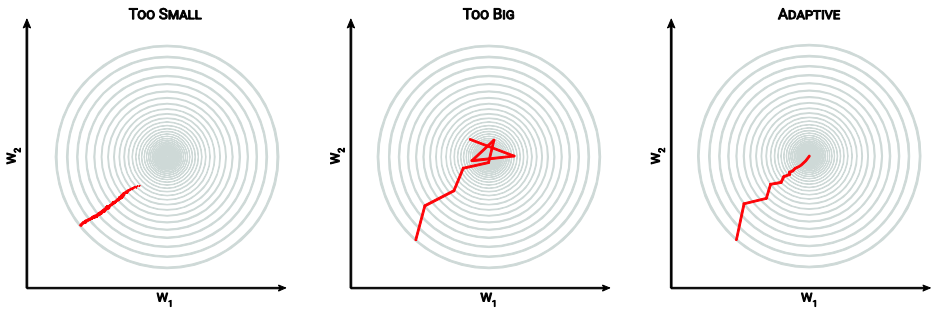


FIGURE 2-21 – CONCEPTUAL ILLUSTRATION SHOWING THE EFFECTS OF LEARNING RATE ON GRADIENT DESCENT

The simplest approach is to use *annealing* [68], which involves decaying the learning rate either according to a pre-defined schedule, or when the cost between epochs falls below a certain threshold. However, these techniques are highly dependent on a priori configuration, making them incapable of dynamically adapting to the characteristics of the dataset [69].

Therefore, several initiatives have been made to make automatically adapting learning rates possible. Among the most successful ones are *AdaGrad* and *RMSProp*.

AdaGrad is essentially an adaptive learning rate relative to the "distance traveled" [70] – that is, it accumulates the gradients all the way from the start. This is done by slowly decaying the learning rate, but not necessarily by the same amount in every direction. As a result, it will decay quickly in directions with steep gradients, and the opposite if the gradients are small. The formula for AdaGrad is given by:

$$\boldsymbol{v}_{i+1} = \boldsymbol{v}_i - \frac{\eta}{\sqrt{\sum_t \nabla \boldsymbol{C}^2}} \nabla \boldsymbol{C}$$

RMSProp is an unpublished adaptive learning rate method very similar to AdaGrad [71]. The only difference is that in RMSProp, the decaying factor (in the denominator) is calculated by exponentially decaying the average and not the sum of gradients. Or put another way, it accumulates gradients over a *window* as opposed to *all* gradients from the start. For this reason, RMSProp is capable of more easily adapting to new data with new properties.

INPUT SCALING

Another trick to improve training accuracy, and to decrease the number of iterations required for gradient descent, is *input scaling*. The mathematical derivation of why this is important is somewhat involved [59], however, the principle is easily illustrated. Figure 2-22 shows a possible feature space with two features w_1 and w_2. If both of the features are within the same range [Figure 2-22 (a)], gradient descent – with a fixed step-size – will more easily find the global minimum. However, for the skewed inputs [Figure 2-22 (b)], gradient descent may overshoot and miss the center entirely – perhaps never manage to land exactly in the minimum. The reason for this is because of the fixed step-size: A small step across one axis (w_2) may lead to a big step on the other axis (w_1), relative to the size of the feature space. Scaling the inputs of the different axis to lie within the same range avoids this problem.

This is obviously not as easy to visualize in higher dimensions, but the same rules apply.

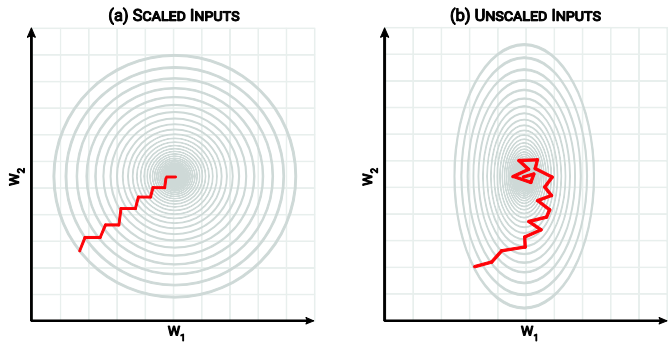


FIGURE 2-22 – CONCEPTUAL GRADIENT DESCENT WITH SCALED AND SKEWED FEATURE VECTORS

2.5.9 REGULARIZATION

Regularization is a common way to reduce overfitting of a dataset in a flexible manner. In principle regularization can be said to be an enforcer of *Occam's razor*[10], by artificially discouraging complex or extreme explanations of the world [50].

The process of regularization is one of many concepts best explained with an illustration. Imagine a dataset split into training and test examples, as shown in Figure 2-23, where each of the dots represents a data point in a two-dimensional feature space. A linear function will do a decent job at approximating the values in the training set, but it is also apparent that the *bias* of the approximation is high – the function *assumes* that the data is linear.

[10] *Occam's razor – https://en.wikipedia.org/wiki/Occam's_razor*

When using the quadratic approximation [Figure 2-23], it becomes evident that it fits the training data well, and the higher order polynomials even more so. However, as the function increases in order it starts to *overfit*. Even though it correctly approximates all of the data points in the training set, the function does a poor job at approximating the test set – it does not generalize beyond the training data.

Nevertheless, plotting the function to see which one fits is not always an option, thus automated regularization techniques are desired. Vastly simplified, regularization solves this by punishing high-valued coefficients more than lower-valued ones, when calculating the error of the approximation – ultimately making the higher order approximation less "extreme", or *smoother,* as seen in the last quadrant in Figure 2-23.

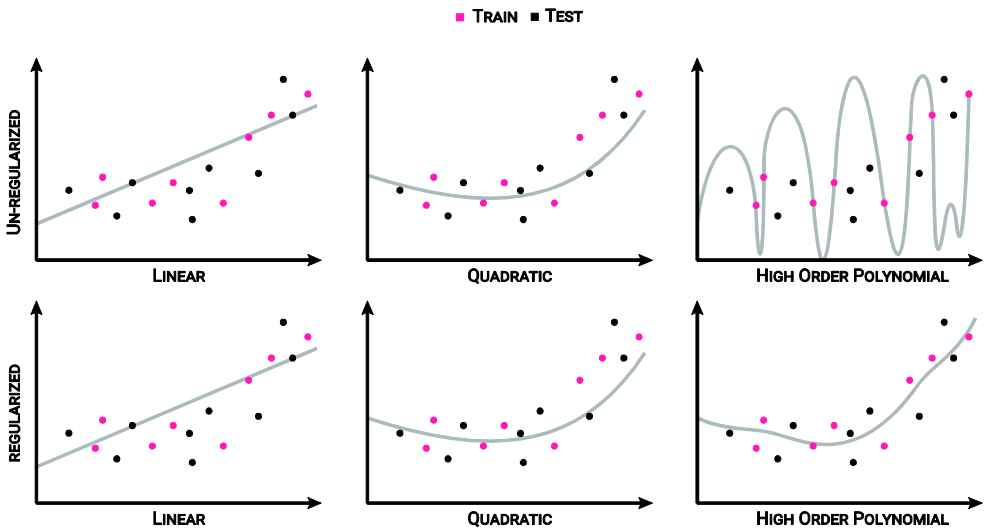


FIGURE 2-23 – CONCEPTUAL REGULARIZATION EXAMPLE

L^2 REGULARIZATION

L^2 regularization, also known as *ridge regression* or *weight decay*, is a penalty parameter that is added to the cost function [53]. The goal of the parameter is to force the model to discriminate towards high-value weights, and instead prefer smaller weights. This makes it harder for local noise to leave a footprint in the model's behavior, thus reducing overfitting.

The actual mathematical term used for regularization varies, but one version commonly used with the CEE cost function (2.4) is defined as [50]:

$$L^2 = \frac{\lambda}{2m}\sum_i |w_i|^2$$

Here λ is a parameter that controls how much the regularizer should affect the cost, and w are the weights of the network. That is, the bias terms are not regularized, as this could lead to *underfitting* [53] – it becomes too general, and incapable of learning.

Making the regularized cost function:

$$CEE = -\frac{1}{m}\sum_{x=1}^{m}[y_x\, ln(a_x) + (1 - y_x)\; ln(1 - a_x)] + \frac{\lambda}{2m}\sum_{i}|w_i|^2 \tag{2.10}$$

Currently, L^2 regularization is one of the standard anti-overfitting techniques in the neural toolkit.

DROPOUT

Dropout is a regularization method that relies on randomly skipping neurons along with their related weights during the training phase [72]. When combined with SGD and backpropagation, dropout can be implemented in such a way that the training algorithm will run the approximation on all of the "thinned" networks in a mini-batch, followed by averaging the gradient for each parameter. Hence, applying dropout is analogous to averaging over multiple networks, which overfits in different ways.

The method is illustrated in Figure 2-24, where the gray nodes denote nodes that are ignored for the given network. By first training two different sub-networks that overfit to a *subset* of the data points, an overfit to the full dataset can be avoided by creating a new network which parameters are the average of the overfitted ones.

Dropout does, however, require some caution when applied to RNNs, as dropping along the *time* axis has shown to degrade performance [73].

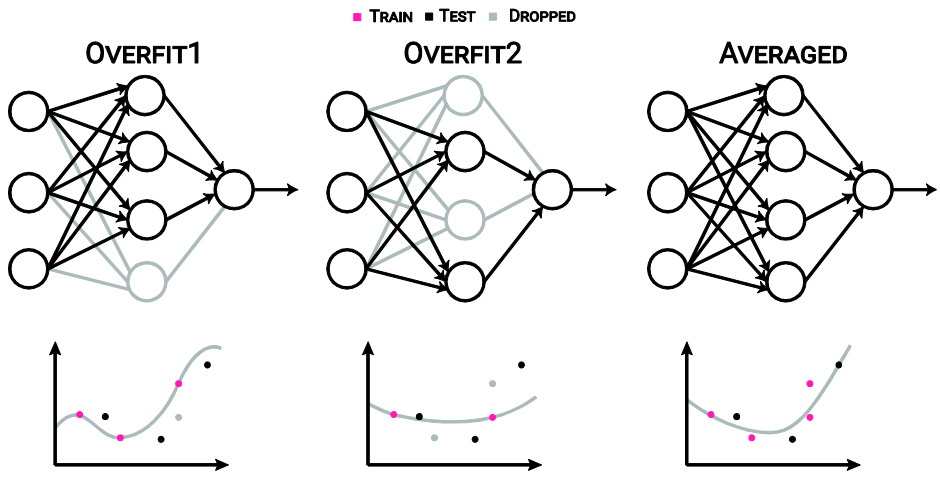


FIGURE 2-24 – CONCEPTUAL ILLUSTRATION OF HOW DROPOUT AVERAGES AND PREVENTS OVERFITTING

EARLY STOPPING

When training an ANN, the number of epochs required in order to reach the sweet spot between overfitting and test set accuracy, is more or less subject to trial and error. *Early stopping* is a technique to automate the choice of epochs, with the help of some simple heuristics [Figure 2-25].

The process is relatively straightforward: After every epoch, evaluate the ANN on the test set. If the error is less than the previous epoch, store the model and run through another epoch. If it is worse, use the currently best-performing network.

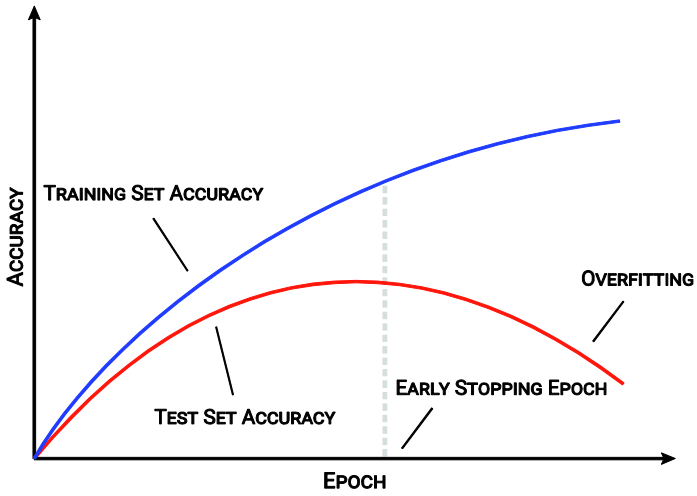


FIGURE 2-25 – EARLY STOPPING

2.6 DIMENSIONALITY REDUCTION

When working with BoW, or other representations whose feature space quickly escalates, it is sometimes desirable to apply techniques to reduce the dimensionality in favor of computability [10], while also sometimes reducing overfitting [5]. These methods are generally split into two categories: *feature selection*, and *feature extraction* [74].

The idea behind feature selection is to select and subset of the original features which provide the most information. For example, by removing all phrases in a corpus which are only used in a single document, as these phrases would be of little help when classifying other documents. Feature extraction, on the other hand, revolves around transforming the original features into another lower-dimensional space. Unlike feature selection, the variables of the resulting space often lose its meaning in the intuitive sense.

The following section covers a brief dig into the domain of dimensionality reduction, including a short note about how it may help understanding the dataset through visualization.

2.6.1 FEATURE EXTRACTION

Principal component analysis (PCA) is feature extraction technique introduced as a mechanism to handle this issue, by attempting to *squeeze* the column space of the vectors together, while still maintaining as much information as possible [39].

The way this is done is by executing a statistical procedure that exploits the spatial relationships by identifying the closest correlated hyperplane, proceeded by projecting every vector onto said plane. By repeatedly applying this process, the PCA algorithm is able to effectively reduce the dimensionality space by one dimension for every iteration.

Perhaps more intuitively, using PCA can be seen as finding the perspective that spreads out the data points the most – thus, capturing the most variance and information possible, for a given dimension. Figure 2-26 illustrates how a set of correlated features in a 2D plane may be projected onto a single dimension.

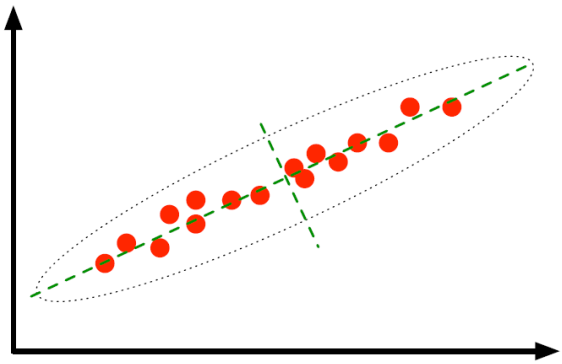


FIGURE 2-26 – PCA ALGORITHM SHOWING THE MAPPING FROM 2 TO 1-DIMENSIONAL SPACE

2.6.2 FEATURE SELECTION

While the computational demands for feature extraction – especially PCA [75] – can be quite demanding, feature selection is often both easier to perform and lighter on the requirements. As touched upon in the introductory section, one way to perform feature selection is to create an upper and lower threshold for the number of occurrences for every phrase (i.e. term frequency filtering). However, this should not be applied blindly, as it has been shown that even rare occurrences might carry significant meaning [31].

Another, yet similar method, is to use a *low-variance* filter. The idea behind this technique is to prune variables with – as the name suggests – low variance. Put another way, given a matrix representation of a dataset (i.e. one vector for every example), low-variance filter removes variables whose data columns' variance lies beneath some threshold.

High-correlation filters are yet another useful feature selection method, based on identifying variables which behave similarly. If two sets of variables are heavily correlated

– they move "up and down" in sync – this technique is based on the assumption that keeping only one of the variables will suffice for the learning algorithm. This can be done by using the *Chi-Square statistic* (χ_2), which measures the degree of association between a term and its corresponding category. Its application is based on the expectation that a term whose frequency strongly depends on the category will be important to account for when discriminating among them [10].

2.6.3 VISUALIZATION

While performing dimensionality reduction to promote computability and otherwise improving the classification process, the reduction may also serve as a visual tool to better understand the dataset. Furthermore, dimensionality reduction can potentially provide a lot of traction in understanding ANNs – which typically works with data in high-dimensional planes [76].

As for the actual techniques, the most prominent in terms of visualization is *t-Distributed Stochastic Neighbor Embedding* (t-SNE) [77]. While mathematically incomprehensible for the common people, the output of the algorithm is quite intelligible. Though, as many related subjects, it is best explained with some visual aid. Figure 2-27 shows a figure from the original t-SNE paper, and depicts a 2D representation of how the classic MNIST dataset (a famous benchmark for recognizing handwritten digits [78]). Although slightly mangled, a closer look reveals that the image contains clusters of similar numbers, with gradual transitions between them. While the axis, nor their values within carry much intuitive meaning, it is the topological structure – the relative position of the numbers – that conveys the message.

Concretely, the plot shows that "1" (bottom right) is the furthest away from "5" (top left), which implies that these numbers are also probably the least related in terms of their original 28x28 pixel space.

Similarly, the same procedure can be used to model news articles and their categories: Both to get a better view of how categories are related, and to identify overlapping relationships. This could then be further utilized to merge very similar categories, or be used as an incentive to investigate and apply more complex methods for those categories in order to better separate them – ultimately uncovering hidden relationships and features of the dataset.

Paraphrased a little more philosophically: Just as there are sounds we cannot hear, and light we cannot see, there might be thoughts we cannot think. Thus, dimensionality reducing tools are created to fill this gap, in an attempt to aid our understanding as a whole.

FIGURE 2-27 – T-SNE PLOT OF THE MNIST DATASET

2.7 TRADITIONAL EVALUATION METHODS

Before delving into the task of the more sophisticated and domain relevant multi-label evaluation methods, a few fundamental concepts related to text classification need to be defined. The following section covers a brief introduction to some traditional evaluation metrics, with a focus on single-label classification.

2.7.1 CONTINGENCY TABLE

A contingency table is a popular way to illustrate several key concepts in information retrieval [5] [79] [80]. Consider a system with n entries, with relationships as shown in Table 2-2. The entries in the table are defined as follows:

$$TP = true\ positive \qquad FN = false\ negative$$

$$FP = false\ positive \qquad TN = true\ negative$$

Each cell in the table specifies the number of times an action took place. For instance, if the system successfully classified five entries as correct, while misclassifying two correct entries as incorrect, you would get a $TP = 5$, and a $FN = 2$.

	YES IS CORRECT	NO IS CORRECT	
DECIDES YES	TP	FP	$TP + FP$
DECIDES NO	FN	TN	$FN + TN$
	$TP + FN$	$FP + TN$	$TP + FP + FN + TN = n$

TABLE 2-2 – CONTINGENCY TABLE

Based on these relationships, three basic evaluation metrics emerge: *recall*, *precision* and *fallout*. Intuitively, recall measures the fraction of relevant entries retrieved, while precision is the fraction of retrieved entries that is relevant. Fallout, on the other hand, is the portion of *non-relevant* entries retrieved.

$$Recall = \frac{TP}{TP + FN} \qquad Precision = \frac{TP}{TP + FP} \qquad Fallout = \frac{FP}{FP + TN}$$

Additionally, a fourth measure *overlap* is sometimes used, which is a symmetric measure with response to FP and FN, which can be used to illustrate how much two categories are alike, disregarding any measure of correctness.

$$Overlap = \frac{TP}{TP + FP + FN}$$

Although the concepts of precision and recall yield valuable outputs on their own, a certain harmony between the two is often desired. Two popular measures for this is *accuracy* and *F-score*:

$$Accuracy = \frac{TP + TN}{n}$$

$$FScore = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$

While the contingency model arguably is one of easier models to grasp, it has two potential drawbacks [80]. The first is that it treats every result with equal importance. Incorrectly labeling a *golf* article as *sports* would have the same impact as labeling *golf* as *gossip*. The second limitation is that it requires all decisions to be binary, thus not suitable for multi-label classification in its pure form.

2.7.2 MICRO- AND MACRO-AVERAGING

When dealing with many labels, averaging measures may be used to obtain a unified evaluation across all the labels. Among approaches applied in literature is micro- and macro-averaging [5] [30] [80]. A macro-average is to calculate the binary classification metrics, like precision and recall, individually for every label, and then apply an averaging measure over them. On the contrary, a micro-average is simply to calculate a single binary classifier for the entire dataset. To get a more formal understanding of the averaging procedures, consider the set of labels $y \in \mathcal{Y}$ and the definitions from Table 2-2. The micro- and macro-averaged binary metrics then becomes:

$$Recall_{micro} = \frac{\sum_{i=1}^{|\mathcal{Y}|} TP_i}{\sum_{i=1}^{|\mathcal{Y}|} (TP_i + FN_i)} \qquad Precision_{micro} = \frac{\sum_{i=1}^{|\mathcal{Y}|} TP_i}{\sum_{i=1}^{|\mathcal{Y}|} (TP_i + FP_i)}$$

$$Recall_{macro} = \frac{\sum_{i=1}^{|\mathcal{Y}|} \frac{TP_i}{TP_i + FN_i}}{|\mathcal{Y}|} \qquad Precision_{macro} = \frac{\sum_{i=1}^{|\mathcal{Y}|} \frac{TP_i}{TP_i + FP_i}}{|\mathcal{Y}|}$$

Intuitively, micro-averaging simply denotes a weighted average based on the number of occurrences for every given category, while macro-averaging weighs every category equally.

Whether micro- or macro is more appropriate depends on the purpose of the categorization. It is argued that micro-average is more applicable when the categories are of similar size, while macro-average, on the other hand, is preferred when classifying documents into equally *important* divisions [80] – however, the optimal choice often requires a call of judgment based on the underlying scenario.

2.8 MULTI-LABEL EVALUATION

While classical evaluation techniques, like the ones presented in Section 2.7.1, are limited to single-label evaluation, multi-label variants carry out the task of multi-label evaluation. This is an important distinction, especially in the news article domain, as articles typically have several *correct* classifications.

There are several useful measures available [81], generally divided into two main categories [30]: *Example-based* and *Label-based.* Example-based metrics starts by evaluating the learning systems' performance on each test example separately, followed by returning some *mean value* based on the test set. Examples of example-based metrics are *subset accuracy* and *hamming loss.* Label-based metrics, on the other hand, evaluates each separate *label* and then returns the micro- or macro-averaged value across every label.

The following section outlines some of the more common evaluation methods. Mathematical definitions used throughout this section are summarized in Table 2-3.

NOTATION	DESCRIPTION
$\mathcal{X}$	d-dimensional instance space $\mathbb{R}^d$
$\mathcal{Y}$	Label space of size q with labels $\{y_1, y_2, \dots, y_q\}$
x	d-dimensional feature vector $\{x_1, x_2, \dots, x_d\}$
Y	Label set associated with x, $Y \in \mathcal{Y}$
$\mathcal{D}$	Multi-label training set $\{(x_i, Y_i) \mid 1 \leq i \leq m\}$
$\mathcal{S}$	Multi-label test set $\{(x_i, Y_i) \mid 1 \leq i \leq p\}$
$h(*)$	Multi-label classifier $h: \mathcal{X} \rightarrow 2^{\mathcal{Y}}$, where $h(x)$ returns the set of proper labels for x

TABLE 2-3 – MATHEMATICAL DEFINITIONS

2.8.1 LABEL CARDINALITY, DENSITY AND DIVERSITY

Before introducing the multi-label evaluation metrics, a handful of convenient measures for general data analysis are elaborated. These measures primarily provide information about the label *structure* rather than evaluating the classification process. Although trivial concepts, these simple analytical tools still give meaningful guidance when exploring the properties of the dataset as a whole.

First off is *label cardinality*, which measures the degree of multi-labelness in the dataset. This is done by calculating the average number of labels per example [30]. The simple nature label cardinality makes it very interesting when characterizing the dataset, as it gives insight regarding the number of categories one should expect from the classifier. The formula for label cardinality is as follows:

$$LCard(\mathcal{D}) = \frac{1}{m}\sum_{i=1}^{m}|Y_i|$$

Label cardinality can be normalized by the total number of different labels, which becomes *label density*:

$$LDen(\mathcal{D}) = \frac{1}{|\mathcal{Y}|} * LCard(\mathcal{D})$$

Another popular multi-labelness measure is *label diversity* [30], which calculates the number of distinct label-sets appearing in the dataset. This gives an intuition of the diversity of the labels themselves. Label diversity is calculated as follows:

$$LDiv(\mathcal{D}) = |\{Y \exists\, x : (x, Y) \in \mathcal{D}\}|$$

Similar to label cardinality, the diversity can be normalized by dividing by the number of examples to indicate the proportion of distinct labels:

$$PLDiv(\mathcal{D}) = \frac{1}{|\mathcal{D}|} * LCard(\mathcal{D})$$

2.8.2 EXAMPLE-BASED EVALUATION METRICS

SUBSET ACCURACY

Subset accuracy is an example-based metric that measures the proportion of correctly classified examples – that is, whether or not the predicted label set is identical to that of the test example. Although a useful measure, it has been criticized for being overly strict when the label space is large [30]. The metric is calculated as follows:

$$SubsetAcc(h) = \frac{1}{p}\sum_{i=1}^{p} |h(x_i) = Y_i|$$

HAMMING LOSS

Hamming loss is another example-based metric for evaluating how many times an instance label pair is misclassified [43] – namely, the fraction of wrong labels with respect to the total number of labels. The equation for calculating the Hamming loss is as follows:

$$HLoss(h) = \frac{1}{p}\sum_{i=1}^{p} \frac{|h(x_i)\Delta Y_i|}{h(x_i) \cap Y_i}$$

where Δ is the symmetric difference between the label sets. The magnitude of the measure evaluates the performance where $hloss(h) = 0$ denotes a *perfect* score (i.e. no misclassified labels).

PRECISION, RECALL, AND ACCURACY

As elaborated in Section 2.7.1 *precision, recall, accuracy* and *f-score* are not very useful when faced with a multi-label evaluation problem. Luckily, these measures all have been formalized for an arbitrary number of labels as well [30]. Although slightly more complicated at first glance, these multi-label variants convey the same message as their simpler single-label cousins. The example-based versions of these metrics are formalized as:

$$Precision_{example}(h) = \frac{1}{p}\sum_{i=1}^{p} \frac{|Y_i \cap h(x_i)|}{|h(x_i)|}$$

$$Recall_{example}(h) = \frac{1}{p}\sum_{i=1}^{p} \frac{|Y_i \cap h(x_i)|}{|Y_i|}$$

$$Accuracy_{example}(h) = \frac{1}{p}\sum_{i=1}^{p} \frac{|Y_i \cap h(x_i)|}{|Y_i \cup h(x_i)|}$$

2.8.3 LABEL-BASED EVALUATION METRICS

Precision, recall, accuracy and F-score, as well as the concepts of micro and macro-averaging [2.7.2] can also be modified for a label-based environment. However, in order to do this, the definitions within the contingency table [Table 2-2] – i.e. TP, TN, FP and FN – need to be defined more generally [30].

Based on the classifier $h(*)$ the multi-label definitions of TP, TN, FP and FN for the j-th class label y_j becomes:

$$TP_j = \left|\{x_i | y_j \in Y_i \wedge y_j \in h(x_i), 1 \leq i \leq p\}\right|$$

$$FP_j = \left|\{x_i | y_j \notin Y_i \wedge y_j \in h(x_i), 1 \leq i \leq p\}\right|$$

$$FN_j = \left|\{x_i | y_j \in Y_i \wedge y_j \notin h(x_i), 1 \leq i \leq p\}\right|$$

$$TN_j = \left|\{x_i | y_j \notin Y_i \wedge y_j \notin h(x_i), 1 \leq i \leq p\}\right|$$

These new definitions can then be used to redefine precision, recall, accuracy and F-score in terms a *label-based* metric:

$$Precision_{label}(TP_j, FP_j, TN_j, FN_j) = \frac{TP_j}{TP_j + FP_j}$$

$$Recall_{label}(TP_j, FP_j, TN_j, FN_j) = \frac{TP_j}{TP_j + FN_j}$$

$$Accuracy_{label}(TP_j, FP_j, TN_j, FN_j) = \frac{TP_j + TN_j}{TP_j + FP_j + TN_j + FN_j}$$

$$FScore_{label}(TP_j, FP_j, TN_j, FN_j) = \frac{2TP_j}{2TP_j + FP_j + FN_j}$$

Then, by letting $B(TP_j, FP_j, TN_j, FN_j)$ represent one of the classification metrics above, micro- and macro-averaging becomes:

$$R_{macro}(h) = \frac{1}{q}\sum_{j=1}^{q} B(TP_j, FP_j, TN_j, FN_j)$$

$$R_{micro}(h) = B(\sum_{j=1}^{q} TP_j, \sum_{j=1}^{q} FP_j, \sum_{j=1}^{q} TN_j, \sum_{j=1}^{q} FN_j)$$

3 RELATED WORK

The amount of research related to text classification is astonishing, and keeping track of it all is an ambitious task. This chapter aims to present a broad, yet concise overview of the *state-of-the-art* in text classification, with emphasis on the news article domain. The priority of the chapter is to serve as a non-technical summary, complementing the more thorough elaboration covered in Chapter 2. Furthermore, it should be noted that this chapter is not an exhaustive source of inspiration for this thesis, but more a fundament that motivated and shaped the remaining chapters.

3.1 THE CURRENT STATE OF MULTI-LABEL LEARNING

The current research on multi-label learning remains in a preliminary state, with most studies in the literature having focused on single-label classification problems [82] [29]. However, due to its omnipresence in real-world problems, it is still an active area of research.

Up until now, the most common approach to multi-label classification, has been to decompose the problem into a set of binary classification problems [34] [29] [30] – with one classifier for each label. A study using a multi-label Naïve Bayes classifier [34], shows that binary decomposition achieves highly competitive performance with respect to specialized multi-label classifiers like the popular AdaBoost.MH [83] and *Multi-Label k-NN* (ML-kNN) [43]. The motivation for using binary decomposition rather than specialized methods, has commonly been because of its computational efficiency and conceptual simplicity [34] [29].

Although simple by virtue, the binary decomposition strategy has received criticism for ignoring correlational information between categories. One study tried to answer these claims using an ML-kNN clustering technique, with experimental results showing *superior performance* over conventional methods [43]. The study showed that using a specialized kNN with simple distance metrics like *Euclidean distance* is a very capable approach in several use cases, spanning from yeast gene functional analysis to natural scene- and web page classification.

3.2 SEMANTICS AND MULTILINGUALISM

The area of focus in recent semantic research has mostly been on exploiting ontological concepts [3] [4] [12]. A common motive for much research has been to challenge the conventional BoW [2.2.1] technique of representing documents – primarily because of its inability to comprehend relationships between important terms that do not occur

literally, as discussed in [3] [10]. Still, there is also conflicting research stating that more sophisticated representations yield little to no improvement over standard BoW [5].

One of the most widespread ontologies used in natural language-related research is *WordNet*[11] [3] [4] [8] [10] [12] [84]. WordNet is an online ontology based on lexical concepts revolving synonyms – or more specifically, *synsets*, which are groups of semantically interchangeable concepts. As a side note: Although WordNet is not used directly in this thesis, it is included here because of its frequent utilization in related studies.

One particular study used WordNet to create an alternative semantic term-weighting-scheme. The study showed that their proposed method outperformed the traditional TF-IDF in cases where the amount of training data was small and the categories well-defined [4]. The main idea behind their approach was to exploit semantic similarity in category labels. However, their results were deemed inconclusive, due to WordNet's incapacity to make sense of proper nouns, like brand names, celebrities, and so on.

Another study used WordNet to create a cross-linguistic classifier by exploiting a graph traversal algorithm to explore and link related and relevant concepts [12]. The resulting method, which consisted of an ensemble of n-binary classifiers, achieved better accuracy over methods using no background knowledge. Both English, Spanish, and Japanese were evaluated – however, since there is no Japanese WordNet, the Japanese dataset was first translated into English before utilized.

The above study is somehow unique in the way that it diverges from the traditional multilingual approach, which often involves (i) creating separate classifiers for every language [85] [86] or (ii) translation into a single language [87] [88]. They also provide evidence that translation alone is insufficient when working in a multilingual domain.

There have also been attempts at using WordNet to manipulate the textual documents directly, as a preprocessing step. One study proposed three main approaches [3]:

(i) *Add related: Text contains "meat" and "salad", add concept "food"*

(ii) *Replace similar: Text contains "tale" and "adventure", replace with "story"*

(iii) *Only use concepts: Ignore all terms not present in the WordNet ontology*

Results revealed a *tiny* fraction of improvement by using ontologies, over the standard binary BoW. Though, the choice of concept strategy was shown to have little impact.

A study with a similar approach criticized some of these techniques showing that incorporating WordNet features, utilizing *part-of-speech* tags during WordNet expansion,

[11] *WordNet – http://wordnetweb.princeton.edu/perl/webwn*

and even term weighting schemes, have little statistically significant positive effect on the accuracy of the Naïve Bayes and SVM classifiers [89]. They used results based on 15 datasets to highlight these claims, ranging from the Reuters news articles to UseNet logs.

On a related note, a team from Google managed to utilize the type information from Freebase[12] entities to surpass state-of-the-art methods using embedding methods for fine-grained entity type classification [90] – that is, they used concrete phrases like "Madonna" as input and predicted multiple labels like "artist", "singer", "woman" and so on. The study shows that there is a significant relationship between entities and their types, which in turn could provide some leverage when clustering entities into more abstract groups of types.

Other research revolving Freebase has often been related to *word sense disambiguation*, and not directly concerned with classification [91] [92]. However, one particularly interesting study attempts to classify blog posts based on extracted named entities [93]. By using off-the-shelf extraction frameworks as a basis, researchers managed to obtain a classification accuracy of 0.69 for a multi-class (not multi-label) classification problem involving seven categories. The methods they used were standard implementations of SVM and Naïve Bayes. Studies like these make Freebase-like ontologies interesting subjects for further research.

However, hierarchical ontologies have also been subject to some concern. One thorough study on the matter disclosed that misclassification on the parent level of an ontology could forcibly misclassify children on a lower level, due to downward error propagation [79]. A conclusion for this particular matter remains unresolved.

3.3 DEEP LEARNING AND WORD2VEC

First and foremost, the amount of related research in the field of deep learning has absolutely exploded over the past decade, with new studies and techniques published on a daily basis. For this reason, the following section covers only some of the highlights, with a focus on recent developments related to RNNs.

Most of the research revolving recurrent nets are based on the LSTM architecture. The areas of application span *anything* that can be represented as a set of ordered inputs. For instance, generating textual descriptions for a wide range of media sources, like speech [94], music [95], images [96] or even video [97] – everything with beyond state-of-the-art accuracy. LSTMs have also been a popular choice for translation because of its *sequence-to-sequence* capabilities [98]. Subsequently, the capabilities of RNNs and deep learning in general have also led to groundbreaking performances in many areas other than

[12] *Another online ontology, discussed further in Section 4.1.4*

computer science. One example is in medicine, where deep learning has been used to predict the activity of drug molecules [99], and analyze the effects of mutations in DNA [100].

In addition to being subject to a variety of academic research, there is also no shortage in more creative areas of application. Andrej Karpathy demonstrated in his blog post[13] that an LSTM network made with only a few hundred lines of Python, could be used to generate anything from Shakespeare and new baby names, to compilable C code. For the latter, he simply trained the network on the Linux kernel, and let the algorithm produce new code based on a seed. Using the same approach, others have been able to generate both humanlike handwriting [101] and composing folk music [95].

Regarding classification, a particularly widespread use for LSTM has been in the domain of sentiment analysis [102] and sequence classification [103] – both with huge success. Especially interesting is the sentiment analysis study, which closely resembles the n-binary multi-label classification problem (each of the binary classifiers is analogous to a sentiment classifier). One of the experiments in the sentiment study [102], used a pre-trained Word2Vec model as the basis for embedded word features, attempting to classify IMDb movie reviews. The possible labels were on a 5-level sentiment scale ranging from *very negative* to *very positive*. The classification was done by splitting the review into paragraphs, and then splitting the paragraphs into sentences. Each of the sentences was then classified, and the results were accumulated on a paragraph level, ultimately making a decision about the full review. The study showed that using Word2Vec features surpassed using standard TF-IDF by an increase of 3.2% in classification accuracy.

Another example of combining deep learning techniques with Word2Vec is a famous study by Yoon Kim using CNNs for sentence classification [17]. He showed that with minimal hyperparameter configuration, a simple convolutional network with only one convolutional layer on top of a feature representation using a pre-trained Word2Vec, could outperform state-of-the-art on several classification tasks. The tasks included both sentiment analysis and sentence classification. The results suggested that the pre-trained Word2Vec models are good and universal feature extractors which can be utilized across datasets.

Finally, another recent study shows that by representing an entire text as a 1D structure – instead of using many low-dimensional word vectors (e.g. Word2Vec) – a CNN similar to those which perform well on 2D images, could outperform the current state-of-the-art in both topic classification and sentiment analysis [104]. The take-home message from the study is that using a single 1D structure, and thus preserve the order of characters in the text, can improve common text classification tasks.

[13] *Unreasonable effectiveness of RNN – https://karpathy.github.io/2015/05/21/rnn-effectiveness/*

In summary, what the current state-of-the-art shows, is that it often does not matter much which areas of application an RNN/LSTM or CNN network is applied to. Extraordinary results have been achieved regardless, with the only requirement being that it is possible to represent the underlying machine learning task as a set of input and output vectors.

3.4 THE NEWS ARTICLE DOMAIN

The news article domain has for a very long time been a particularly interesting and relevant area of focus in text classification, much due to the nature of the content [2]. As touched upon in Section 1.1, news articles have traditionally been classified by hand, which is a tedious and error-prone procedure. This, combined with the fact that news articles often have several matching categories, may lead to a significant amount of ambiguity in the classification process. The takeaway is, not surprisingly, that news classification has been a long awaited task to automate, which ultimately results in an overwhelming amount of related research. The following section attempts to highlight some of the substantial studies deemed relevant for this thesis, with a focus rooted in datasets and classification methods.

3.4.1 CHARACTERISTICS AND COMMON APPROACHES

Formal research characterizing news articles is hard to come by. However, some features can be inferred by taking a quick glance at any commercial newspaper. First and foremost, news articles are mostly written by professionals, and the writing style is typically formal and focused [4]. Second, it is fair to assume that news articles may have very high dimensionality when represented as a BoW. This assumption is based on the notion that news articles often have a high frequency of proper nouns, like names of people, places, countries, organizations and more. This also fits well with the common opinion that text is a media of high dimensionality, with few irrelevant features [38].

By combining these characteristics one may assume that performing news classification in a reliable way is subject to significant computational deficiencies. This supposition also makes sense in a research context, as several popular algorithms used within the domain are based on lightweight classifiers like Naïve Bayes [8] [5] and SVM [38]. SVM has for instance produced promising results while only requiring 20 training examples and 100 iterations to become stable – and still outperform both Naïve Byes and k-NN [79]. Furthermore, Naïve Bayes in its pure form, has been shown to inherit significant bias towards skewed label distributions, giving greater probabilities to high-frequency labels [37].

Regarding classification techniques, it is worth noting that recent research point in the direction that given enough data, the choice of the classifier is no longer as crucial [31]. The hypothesis that motivates this claim is that as the dataset grows, the training data

gradually approaches the true distribution of the underlying data itself – thus, making overfitting a problem of the past.

3.4.2 DATASETS

Most of the research done in the news article domain, or even text classification in general, have traditionally used a variation of the *Reuters-21578*[14] dataset [1] [3] [4] [38] [79] [12]. The dataset is, as implied by its name, a collection of 21578 manually categorized news articles, collected and labeled by the Carnegie Group, Inc. and Reuters, Ltd. Although a valuable dataset, it is vulnerable to criticism for its size – especially for multi-label classification, which by default requires a lot more data. However, over the last decade a new generation of more recent datasets from Reuters have emerged: RCV1 [105], a news article corpus containing 810 000 articles released in 2000, spanning a single year worth of English articles. RCV2, a smaller version containing articles from 13 languages was published in 2005. These datasets have since gained considerable traction in the text classification domain [29] [12] [104] [106].

An even bigger dataset was released in 2008, by *The New York Times* (NYT). They released a massive corpus with more than 1.8 million of labeled articles – a perfect playground for data mining and a genuine resource for classification tasks. Since then, the corpus has become subject to multiple interesting studies, ranging from Twitter analysis [107] and event detection [108] to barebones data mining [109]. There has also been an initiative to improve the categorizations supplied in the corpus [14], but the real potential in the field of classification has been left more or less unexplored.

There are however some exceptions to this. For instance, a recent study by Google which used the corpus to implement a system for extracting Freebase entities (similar to [93]) and assigning relevance score (salience) for each entity in every document [110]. Although the study only used a small portion (100 000 documents) of the dataset, Google showed that features derived from an entity-first approach were more robust than simple word-count features typical of a keyword extraction system. They concluded that there is much potential in this type of approach, and encouraged similar research.

3.4.3 EVALUATION

As with text categorization in general, the most common way to evaluate classification results is by using macro-averaged precision, recall and F-score. F-score being the most significant, with anything over 0.7 often regarded as satisfactory [10] [29] [79] [82]. These observations are in line with the argumentation in Section 2.7.2. Another popular metric is Hamming loss, with acceptable values spanning a range of 0.03 to 0.06 [34]. However,

[14] *Reuters-21578 – http://www.daviddlewis.com/resources/testcollections/reuters21578/*

it is also argued that numbers like these carry little weight, as they are really only comparable when conducting experiments with the *exact same* datasets – motivating that one should rather compare them with the results of a human expert [5] [111].

4 DATA

As touched upon in Chapter 3, the quantity of well-categorized training data has been scarce and hard to come by [4], making the processes of classifying things like news articles a challenging task. However, this inconvenience took a turn when *The New York Times* (NYT) in 2008 released a *massive* corpus to the public[15]. The corpus is composed of manually categorized and handcrafted news articles, spanning twenty years of news history. Containing more than 1.8 million articles, NYT gave life to an enormous potential in text classification, especially in the news article domain. For this reason, the corpus creates a natural data foundation for many classification tasks.

Motivated by Google's preliminary work with the corpus [110] (performing entity extraction and classification as discussed in Section 3.4.2), the NYT corpus was selected as the main dataset of choice for this thesis. Although explained in great detail in Section 4.2, the main idea was to use it as the basis for a language-independent feature extraction process much like done in [93], composed of:

(i) *extracting language-independent entities for each article*

(ii) *and mapping the articles into a language-independent category space*

The purpose of this chapter is to serve as an elaborate presentation on how this process, motivated by findings in Chapter 3, was carried out. This is done by first examining a set of relevant data sources, followed by a thorough demonstration of how they were connected.

4.1 DATA SOURCES

In addition to the NYT corpus, numerous other data sources have been assessed and utilized for this research. The following section contains a brief overview of how and why the data sources were chosen.

4.1.1 NEW YORK TIMES ANNOTATED CORPUS

Before delving into the other data sources, let us take a closer look at the properties of the NYT corpus. As mentioned, the corpus contains more than 1.8 million articles. Out of these, 1.5 million are manually categorized by library scientists, while the remaining are algorithmically tagged under supervision by NYT's production staff. An overview of the article attributes available in the corpus is shown in Table 4-1. The *relevant* column in the table shows which attributes were used for this project. To fully grasp the

[15] *The New York Times Annotated Corpus – https://catalog.ldc.upenn.edu/LDC2008T19*

immensity of the corpus, a number of interesting statistics are displayed in Table 4-2. An example of a full article is available in Appendix A.1.

Needless to say, the NYT corpus is an incredible testbed for text classification tasks rooted in the news article domain, or even text related research in general.

ATTRIBUTE	RELEVANT	ATTRIBUTE	RELEVANT
alternativeURL		onlineDescriptors	x
articleAbstract		onlineHeadline	
authorBiography		onlineLeadParagraph	
Banner		onlineLocations	
biographicalCategories		onlineOrganizations	
Body		onlinePeople	
Byline		onlineSection	
columnName		onlineTitles	
columnNumber		Organizations	
correctionDate		Page	
correctionText		People	
Credit		publicationDate	
Dateline		publicationDayOfMonth	
dayOfWeek		publicationMonth	
Descriptors	x	publicationYear	
featurePage		Section	
generalOnlineDescriptors	x	seriesName	
Guid	x	Slug	
Headline	x	sourceFile	
Kicker		taxonomicClassifiers	x
leadParagraph		Titles	
Locations		typesOfMaterial	
Names		url	x
newsDesk		wordCount	
normalizedByline			

TABLE 4-1 – FULL SET OF ATTRIBUTES THE NYT CORPUS. RELEVANCE DENOTES IF THE ATTRIBUTE IS USED.

DESCRIPTION	VALUE
Article count	1 855 658
Article summary count	650 000 +
Number of taxonomic classifiers	8 300 051
Number of descriptors	9 512 574
Number of attributes per article	49
Zipped size	3 GB
Uncompressed size	15.6 GB
Data format	New Industry Text Format (NITF)
Year span	1987 – 2007

TABLE 4-2 – NYT CORPUS STATISTICS

4.1.2 IPTC AND MEDIA TOPICS

The International Press Telecommunication Council[16] (IPTC) is a consortium of the world's major news agencies and providers, aiming to simplify the distribution of information. With affiliates like *Reuters*, *Associated Press, BBC* and *The New York Times*, the organization acts as *the global standards body* for the news industry.

One of IPTC's major initiatives, the *Media Topics*[17], is an ontology of *news categories*. At the time of writing, the ontology is made up of 1130 different *language neutral* topics, distributed throughout the ontology under 17 top level categories. Each main category holds up to 5 nested levels of increasingly specific sub-categories. Figure 4-1 shows an excerpt of the Media Topics ontology.

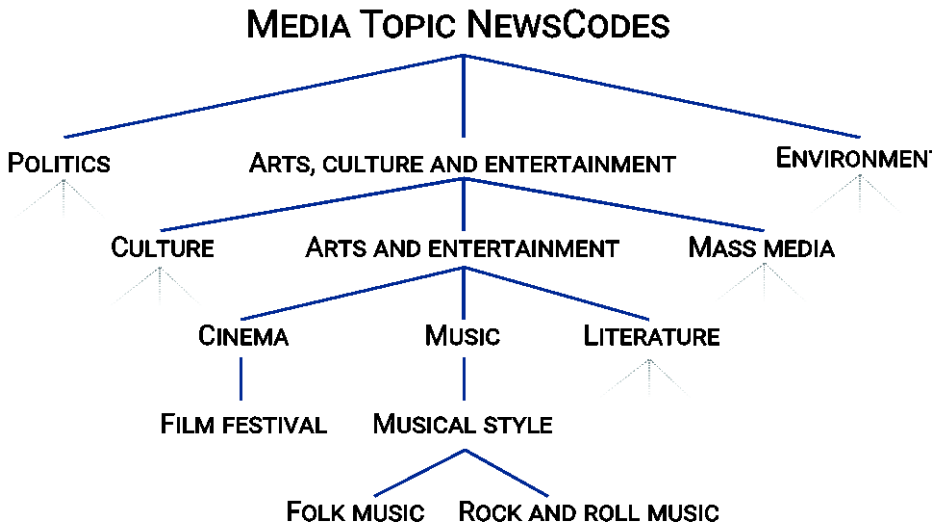


FIGURE 4-1 – MEDIA TOPICS ONTOLOGY EXCERPT

Being subject to constant development, the Media Topics ontology is a vigorous and potentially important building block when creating a language-independent news classifier. The fact that it is maintained by some of the biggest players in the industry, only adds to the robustness of the project, which makes it a natural fundamental guideline for classifying news articles. For these reasons, IPTC Media Topics were chosen as a suitable basis for the category set used in this thesis.

[16] *IPTC – https://iptc.org/*

[17] *IPTC, Media Topics – https://iptc.org/standards/media-topics/*

4.1.3 NEW YORK TIMES TO IPTC

To take advantage of the potential of IPTC's Media Topics, a mapping between the categorical descriptors in the NYT corpus, and matching Media Topics is required. Fundamentally, the process of creating this mapping is a difficult, and not to mention tedious, task. This is mainly because the corpus contains an *overwhelming* number of categories, many of which do not have obvious counterparts in the Media Topic domain. Direct matching by word is not the desired approach, as the category space itself, is subject to severe word sense ambiguities.

For this reason, in order to achieve a sensible mapping between NYT descriptors and corresponding Media Topics, all of the mappings would require individual quality assurance by a domain expert, to certify the link. Lucky, this work has already been done, and with much help from *Evan Sandhaus*[18], the creator of the NYT corpus, the mapping was acquired. An excerpt of the mapping is displayed in Table 4-3.

NYT CATEGORY	SKOS MATCH	IPTC MEDIA TOPIC	ID
Privacy	skos:narrowMatch	data protection	20000627
Presidents Cup (Golf)	skos:broadMatch	golf	20000940
Presidents' Day	skos:broadMatch	public holiday	20000552
Pretzels	skos:broadMatch	food and drink	20000568
Preventive Medicine	skos:exactMatch	preventative medicine	20000476
Prices (Fares, Fees and Rates)	skos:exactMatch	prices	20000382
Priests	skos:broadMatch	religious leader	20000703
Primaries and Caucuses	skos:exactMatch	primary	20000583
Principals (School)	skos:closeMatch	teachers	20000416
Prison Escapes	skos:broadMatch	prison	20000137

TABLE 4-3 – NYT TO IPTC MAPPING EXCERPT

As shown in Table 4-3, the link between NYT category and the Media Topic is trailed with a matching attribute. The matching attribute determines *how* the descriptor and Media Topic is related, and follows the *Simple Knowledge Organization System RDF Schema*[19] ontology, which is a framework for tying and defining related concepts.

[18] *Evan Sandhaus – http://evansandhaus.com/*

[19] *SKOS – http://www.w3.org/TR/2008/WD-skos-reference-20080829/skos.html*

4.1.4 ONLINE ONTOLOGIES

Structured data linked together across language barriers, and abstracted away from grammatical nuances, is key when it comes to labeling news articles in a multilingual setting. Unfortunately, the task of extracting entities is far from trivial. Word disambiguation is already incredibly difficult for a single language – how should one proceed with such a task for multiple languages? Fortunately, this problem has existed for some time, and possible solutions exist. The magic phrase is *online ontologies*: A set of unifying databases where concepts and entities are reduced to language agnostic identifiers, knit together forming graph-like structures of relations. This section explores the online ontologies investigated in this thesis.

FREEBASE

Freebase used to be a large collaborative knowledge base. Ever since its launch in 2007 and until its retirement in 2015 it was an open project maintained by a dedicated community. At the time of its discontinuation, it contained a whopping 3 041 722 635 facts, covering 49 947 845 different topics. The topics range from *celebrities* to *home appliances*, each paired with a unique, *language-independent*, ID – together forming a massive online ontology, relating an astonishing number of diverse concepts and entities.

The reason why such an incredible resource was discontinued might be puzzling to some, however, their motivation was clear: In order to completely fulfill and successfully maintain such an ambitious project, there are not a lot of room for competitors – which leads the introduction of *Wikidata*[20].

WIKIDATA

Wikidata is a project operated by the *Wikimedia Foundation*[21], and is in many ways an identical initiative with respect to Freebase. The major difference is that Wikidata is backed by Wikimedia – an organization already very familiar with handling huge amounts of data. For this reason, the natural outcome for Freebase was to shut down, and mobilize a merge of the two databases, that is, move everything from Freebase into Wikidata.

However, because of Wikidata's strict policy regarding the credibility of sources and overall integrity of *facts*, the mapping from Freebase turned out less than trivial[22].

[20] *Wikidata – https://www.Wikidata.org/*

[21] *Wikimedia Foundation – https://en.wikipedia.org/wiki/Wikimedia_Foundation*

[22] *Freebase's announcement – https://plus.google.com/109936836907132434202/posts/bu3z2wVqcQc*

A lot of the Freebase entities could not be automatically mapped to their corresponding Wikidata concepts, which in turn led to a significant number of entities getting lost in the transformation.

Although the Freebase to Wikidata translation may result in a substantial loss of information, the quality and verbosity of Wikidata still make it a very prominent resource.

DBPEDIA

As mentioned, Wikidata is governed by strict editing policies with a focus on quality rather than quantity. But there is also a third alternative, DBpedia[23] with more or less the opposite. Whereas Wikidata's information is structured natively, and provided by Wikimedia directly from Wikipedia, DBpedia uses an automated approach based on crawling and parsing on its own. For this reason, DBpedia gathers a lot more information and captures more relations than Wikidata – all at the price of being less accurate and more relaxed in terms of structure.

Although a little less organized, DBpedia's real strength stems from its richness. Another selling point is its maturity. DBpedia was launched around the same time as Freebase in 2007, and has been collecting data ever since. Having served almost ten years in service, the ecosystem around DBpedia is also rather diverse and widespread, containing multiple third party tools covering anything from working with the API to automatic entity extractors.

4.1.5 ENTITY EXTRACTORS

Entity extraction is hard. Not only is it hard, but it is also a very computationally expensive and time-consuming process. This is part because of the amount of training required, but also due to the mere complexity of language itself. The process is also often reliant on pre-trained language models in order to gain a high success rate. Building such tools from the ground up is a comprehensive process subject to an entire thesis on its own. For this reason, the focus of this project has been on utilizing publicly available entity extraction and *part-of-speech* tagging software.

Among the extractors investigated were the well-known NLP libraries *OpenNLP*[24] and *Stanford NLP*[25], as well as *DBpedia Spotlight*[26]. Whereas OpenNLP and Stanford NLP provides a quite low-level NLP API, DBpedia Spotlight a high-level extractor built on

[23] *DBpedia – http://wiki.dbpedia.org/*

[24] *OpenNLP – https://opennlp.apache.org/*

[25] *Stanford NLP – http://nlp.stanford.edu/software/*

[26] *DBpedia Spotlight – https://github.com/dbpedia-spotlight/dbpedia-spotlight*

top of several such APIs (including OpenNLP) [112]. DBpedia is also, as the name suggests, tightly integrated with DBpedia, making it possible to extract DBpedia entities directly.

FIGURE 4-2 – DBPEDIA ONLINE DEMO AVAILABLE AT DBPEDIA-SPOTLIGHT.GITHUB.IO/DEMO/

While OpenNLP and Stanford NLP both are well-performing libraries backed by thoroughly trained language models, DBpedia Spotlight became the natural choice for this project due to its ease of use and tight integration with DBpedia.

In addition to being well performant and easy to use, DBpedia Spotlight is also very resourceful with a wide variety of features. An example from the online demo of the annotation API is shown in Figure 4-2, where entities extracted from a sample text are underlined, each accompanied by a link to its corresponding DBpedia page.

There are several customizable parameters for the API, with the most important ones being type and language selection, and confidence. Type selection is related to the online ontologies' *type hierarchy*. For instance, among the types of *Berlin* are *Thing, Place, Location, City* and so on. DBpedia Spotlight is capable of extracting nearly any existing DBpedia type. With other words, the type selector in DBpedia Spotlight is merely a filter for which kind of *things* you want the API to extract.

As for the methodology, DBpedia Spotlight's entity extraction is a two-step process, initialized by first performing entity disambiguation based upon cosine similarities and TF-IDF weights. After the disambiguation is completed, the identification and extraction of entities is just a matter of exact string matching [112].

Furthermore, DBpedia Spotlight supports extraction from a total of ten languages out of the box: English, German, French, and Spanish to mention some. According to the developers, the framework should in principle be extendable to any language that has a Wikipedia edition, as that is the main resource for model training [112].

The last parameter to tune is *confidence* – a measure of how sure the extractor is that it extracted the correct entity. An extractor with a confidence of 0.5 is going to return a higher number of entries, possibly with some false positives, while an extractor with a confidence of 0.9 will return fewer, but more certain entities.

4.1.6 WORD2VEC

As elaborated in Section 2.2.3, word embeddings is an alternative way to represent features in a document as opposed to the traditional BoW approach. However, unlike BoW features, whose construction is merely a matter of counting, the word embeddings need to be trained. Not only that, but in order to obtain a set of word embeddings that closely represents the relationships in the language itself, the required amount of training data is potentially enormous. For this reason, training word embeddings may quickly turn into a very computationally demanding task.

A striking side effect of training and creating a robust set of word embeddings, is that the resulting model also becomes extremely versatile – hence, it can be reused. This leads us to the Google's *Word2Vec* project[27] [16] [113] – an open source toolkit for working with, and training word embeddings. In addition to providing implementations of the algorithms required to train and utilize Word2Vec vectors, the toolkit also includes a number of pre-trained models – some of which are trained with more than 100 billion words from *Google News*. Even more interestingly: One of the models provided is trained for Freebase data. The model contains an almost complete mapping between Freebase ID's and a set of 1000-dimensional Word2Vec vectors.

As for the actual training, Google's Word2Vec vectors are trained with a two-layer neural network using the *skip-gram* method [2.2.3]. The network processes the entire corpus as input, and output a set of Word2Vec vectors.

For the reasons above, Google's pre-trained Word2Vec Freebase model turned out to be the perfect source for the word embedded feature representation for this project.

[27] *Word2Vec – https://code.google.com/archive/p/word2vec/*

4.2 PREPROCESSING

After having gathered a handful of desirable resources, the preprocessing pipeline is initiated to kick start the classification process. For this thesis, the preprocessing of the data is done in five steps, as outlined by Figure 4-3. The first four steps involve joining and filtering the different data sources, while the last step revolves around cleaning up, and computing the feature vectors.

In order to provide the reader with a solid understanding of the entire preprocessing operation, the demonstration of the pipeline is complemented with an illustrative example, using the New York Times article *1821747* as a basis. The full raw article is available in Appendix A.1.

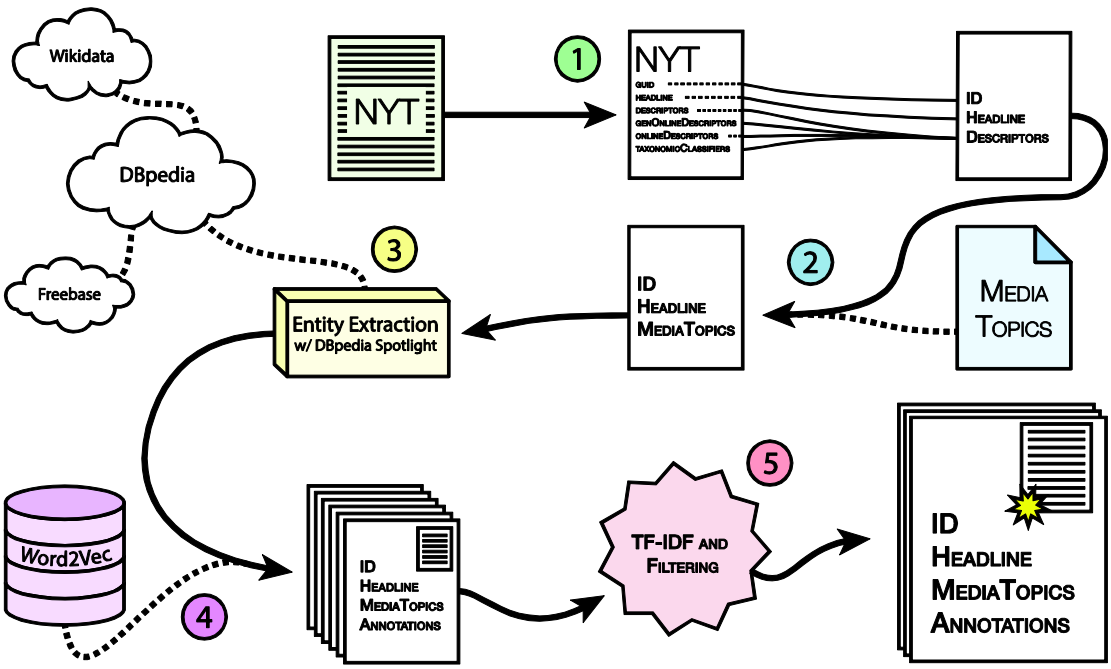


FIGURE 4-3 – PREPROCESSING PIPELINE

4.2.1 THE PROCEDURE

STEP 1 – EXTRACTING THE RELEVANT RAW-DATA

The initial preprocessing step involves loading the corpus from disk, followed by extracting the *relevant* attributes based on Table 4-1. This step reduces the article from Appendix A.1 to the following:

```
id:                          1821747
headline:                    Bush Defends Moving Against Iranians Who Help Shiites Attack U.S.-Led Forces in Iraq
descriptors:                 [Shiite Muslims,
                              United States International Relations,
                              United States Armament and Defense]
generalOnlineDescriptors:[United States International Relations,
                              United States International Relations,
                              Shiite Muslims,
                              International Relations,
                              United States Armament and Defense,
                              Armament,
                              Defense and Military Forces,
                              United States Politics and Government,
                              Politics and Government]
onlineDescriptors:           [United States Armament and Defense]
taxonomicClassifiers:        [Top/News,
                              Top/News/World/Countries and Territories/Iran,
                              Top/News/World/Countries and Territories/Iraq,
                              Top/News/World,
                              Top/News/Washington,
                              Top/News/Washington/Campaign 2004/Candidates,
                              Top/News/World/Middle East,
                              Top/Features/Travel/Guides/Destinations/Middle East,
                              Top/Features/Travel/Guides/Destinations/Middle East/Iran,
                              Top/Features/Travel/Guides/Destinations/Middle East/Iraq]
url:                         http://query.nytimes.com/gst/fullpage.html?res=9C01EFDB173FF934A15752C0A9619C8B63
```

FIGURE 4-4 – RAW NYT ARTICLE WITH RELEVANT ATTRIBUTES

The most notable attributes at this point are *generalOnlineDescritors, onlineDescriptors* and *taxonomicClassifiers.* However, as seen in Figure 4-4 the categorical data is spread across four different attributes, in two different textual formats. One format being a list of *descriptors* whilst the latter is a *taxonomy* (ontology). There is also a lot of duplication. A quick and easy fix for this subtle inconvenience is to split the taxonomic entries on the "/" character, extracting the *last* descriptor, and then take the union of all of the descriptive categories combined. The result is trimmed down article representation with a single *descriptors* category, as shown in Figure 4-5:

```
id:          1821747
headline:    Bush Defends Moving Against Iranians Who Help Shiites Attack U.S.-Led Forces in Iraq
descriptors: [united states international relations, world, politics and government,
              united states armament and defense, candidates, middle east, armament,
              defense and military forces, iran, international relations, washington,
              united states politics and government, shiite muslims,
              countries and territories, iraq, campaign 2004]
url:         http://query.nytimes.com/gst/fullpage.html?res=9C01EFDB173FF934A15752C0A9619C8B63
```

FIGURE 4-5 – RAW NYT ARTICLE WITH COMBINED DESCRIPTORS

STEP 2 – MAPPING DESCRIPTORS TO IPTC MEDIA TOPICS

The second preprocessing step is mapping the categorical information to IPTC Media Topic categories according to the relationships in Table 4-3. Mapping the NYT descriptors to Media Topics is arguably the most crucial step of the entire pipeline. This is because the mapping makes it possible to create an NYT-independent link between the article contents and its category, and thus facilitating language neutrality. However, the process is also a very unforgiving, as a lot of the descriptors do not have any matches. This is because the mapping only contains very general terms. For instance, anything related to geographical areas, people or otherwise national descriptors, will not have any match in Media Topics. In fact, only general descriptors like *politics*, *family* and *religion and belief* will have matching counterparts in Media Topics.

As shown in Table 4-3, the mapping is in principle straightforward: If the NYT descriptor has a match in Media Topics, *add it*, otherwise *ignore*. However, to reduce the problem domain, two different types of mapping mechanisms were attempted: *Best* and *Broad*.

While *Best* simply maps to the *closest* Media Topic, *Broad* – as the name suggests – maps to the *broadest* Media Topic. The latter is done by initially selecting the closest match, and then traversing the Media Topic ontology upwards until reaching the top level. Using a broad map gives a clear advantage with regard to the complexity of the classification process, as the number of top level categories is orders of magnitude lower than the entire category set – from 1130 to 17 at the time of writing. When using an n-binary multi-label classifier, this means the number of models to train, is reduced from 1130 to 17.

An example of the two NYT to Media Topic mappings for the example article is depicted in Figure 4-6 (Broad) and Figure 4-7 (Best).

```
united states international relations   -> politics
politics and government                 -> politics
international relations                 -> politics
united states politics and government   -> politics
shiite muslims                          -> religion and belieif
defense and military forces             -> politics
```

FIGURE 4-6 – NYT --> IPTC MAPPING, BROADEST

```
united states international relations   -> international relations
politics and government                 -> politics
international relations                 -> international relations
united states politics and government   -> politics
shiite muslims                          -> Islam
defense and military forces             -> defence
```

FIGURE 4-7 – NYT --> IPTC MAPPING, BEST

Moving on with the *Broad* mapping, the example article transforms into:

```
id:         1821747
headline:   Bush Defends Moving Against Iranians Who Help Shiites Attack U.S.-Led Forces in Iraq
iptc:       [politics, lifestyle and leisure, religion and belief]
url:        http://query.nytimes.com/gst/fullpage.html?res=9C01EFDB173FF934A15752C0A9619C8B63
```

FIGURE 4-8 – NYT ARTICLE AFTER IPTC MAPPING

STEP 3.1 – ENTITY EXTRACTION

Once the categories are mapped, the next step in the pipeline is to perform entity extraction and create annotations for each article. The raw entity extraction is done through an offline version of DBpedia Spotlight[28] (v0.7), and its REST API. As mentioned in Section 4.1.5, there are three main parameters to consider when using the API. The language selection is given, as the NYT corpus is exclusively in English. As for the types, *all* types are chosen, to closer represent the contents of the article. The confidence level was initially set to the default value of 50 %, as this seemed to give a decent balance between too many, and too few annotations. A later experiment [6.1] explores the impact of the different confidence levels more thoroughly.

As the actual article *1821747* is quite long (997 words), Figure 4-9 shows a snippet used to illustrate the differences between some of the available confidence levels. Figure 4-10 depicts the raw data extracted with the DBpedia Spotlight REST API with confidence 50 %. The first value is the DBpedia ID, which corresponds to the lexical suffix of any Wikipedia URL (e.g. http://en.wikipedia.org/wiki/Zalmay_Khalilzad). The second value is the offset – that is, the location of the entity within the document, by character. The third value is a set of *types*, which are elaborated upon in Step 3.2.

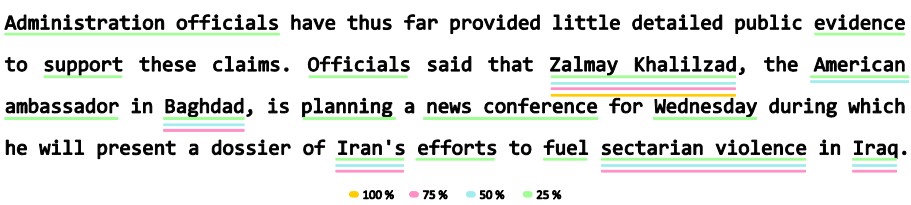


FIGURE 4-9 – CONFIDENCE LEVELS EXAMPLE

```
id:      Zalmay_Khalilzad > offset:  3106 > types: Q215627, Q5
id:              Baghdad > offset:  3151 > types: Q486972, Q515
id:      Politics_of_Iran > offset:  2643 > types:
id:          Sectarianism > offset:   296 > types:
id:    Sectarian_violence > offset:  3311 > types:
id: 2003_invasion_of_Iraq > offset:  3333 > types: Q1656682
```

FIGURE 4-10 – ENTITIES EXTRACTED WITH DBPEDIA SPOTLIGHT

As shown in Figure 4-10, the entities extracted from DBpedia have does not necessarily have IDs matching the phrase that triggered the extraction. For instance, *Iraq* was identified as *2003_invasion_of_Iraq*. As touched upon in Section 4.1.5 this is due to DBpedia Spotlight's word disambiguation schemes, which attempts to find entities based on its surrounding context.

[28] *DBpedia Spotlight v1.2.7 – http://spotlight.sztaki.hu/downloads/*

Another remark is that the extracted ID is in the DBpedia format, which is incompatible with the Freebase (/m/<id>) and Wikidata (Q<id>) formats. However, as the DBpedia ID corresponds to the lexical Wikipedia ID, the Wikidata ID can be extracted from the full Wikidata ontology – given that the entity actually exists in the other ontologies, that is.

The actual extraction of the Wikidata ID can be done either by downloading and parsing an offline Wikidata dump, or through the *Wikidata Query API*[29]. The same can be done for the Freebase ID, which is also present on Wikidata, as a result of the attempt of incorporating Freebase's knowledge base into Wikidata after Freebase's discontinuation [4.1.4].

It should be mentioned that the Freebase to Wikidata translation is not optimal – much because of the differing policies and other difficulties related to the Freebase to Wikidata transferring process, which ultimately led to a significant number of Freebase entities getting lost in transit, as touched upon in Section 4.1.4. However, for the annotations in the example, the mapping was quite successful, and a match was acquired for all Freebase ID's. The result of the translation process is depicted in Figure 4-11.

```
     Zalmay_Khalilzad > fb: /m/01bjnp > wb:  Q145193 > mc:  1 > offset: 3106 > types: Q215627, Q5
              Baghdad > fb:  /m/01fqm > wb:    Q1530 > mc:  3 > offset: 3151 > types: Q486972, Q515
     Politics_of_Iran > fb:  /m/0h7zd > wb: Q1031246 > mc: 13 > offset: 2643 > types:
         Sectarianism > fb: /m/023kmz > wb: Q2352121 > mc:  1 > offset:  296 > types:
   Sectarian_violence > fb: /m/02_drp > wb: Q3774758 > mc:  1 > offset: 3311 > types:
2003_invasion_of_Iraq > fb: /m/01cpp0 > wb:  Q107802 > mc: 16 > offset: 3333 > types: Q1656682
```

FIGURE 4-11 – EXTRACTED ANNOTATION AFTER PROCESSING

In addition to the merged IDs, the translation also introduced a value *mc* which is the *mention count* for the given annotation within the text. That is, the number of occurrences that the entity appears within the document. For simplicity, only the first occurrence is captured in the annotation list attached to the document, however the mention count is required to compute the TF-IDF [2.2.2] later on.

STEP 3.2 – INCORPORATING TYPES (OPTIONAL)

Once all of the annotations from DBpedia Spotlight have been merged into the articles, there is one optional step left. This step involves incorporating the *types* returned from DBpedia, as shown in Figure 4-11, into the article in addition to the other entities. In short the types can be seen as the *generalization,* or superclass of an entity. For instance, the entity *Zalmay_ Khalilzad* (Q145193) has the types *Person*[30] (Q215627) and *Human*[31] (Q5). See Section 2.1.2 for more information about ontologies.

[29] *Wikidata Query API – https://wdq.wmflabs.org/api_documentation.html*

[30] *Person – https://www.Wikidata.org/wiki/Q215627*

[31] *Human – https://www.wikidata.org/wiki/Q5*

The act of padding articles with types is just a matter of translating the types into "normal" annotations, and add them the normal way. For the purpose of simplicity, the mention count and offset of the new type-annotation are chosen to be the same values as those of the annotation it was created from. Figure 4-12 shows the list of annotations for the running example with types incorporated. *NONE* means that there was no matching Freebase ID for that particular Wikidata entity.

```
      Zalmay_Khalilzad > fb: /m/01bjnp > wb:  Q145193 > mc:  1 > offset: 3106 > types: Q215627, Q5
               Baghdad > fb:  /m/01fqm > wb:    Q1530 > mc:  3 > offset: 3151 > types: Q486972, Q515
      Politics_of_Iran > fb:  /m/0h7zd > wb: Q1031246 > mc: 13 > offset: 2643 > types:
          Sectarianism > fb: /m/023kmz > wb: Q2352121 > mc:  1 > offset:  296 > types:
    Sectarian_violence > fb: /m/02_drp > wb: Q3774758 > mc:  1 > offset: 3311 > types:
 2003_invasion_of_Iraq > fb: /m/01cpp0 > wb:  Q107802 > mc: 16 > offset: 3333 > types: Q1656682

          type->Person > fb:      NONE > wb:  Q215627 > mc:  1 > offset: 3106
           type->Human > fb: /m/0dgw9r > wb:       Q5 > mc:  1 > offset: 3106
type->Human_settlement > fb: /m/0fpdgl > wb:  Q486972 > mc:  3 > offset: 3151
            type->City > fb:      NONE > wb:     Q515 > mc:  3 > offset: 3151
           type->Event > fb:      NONE > wb: Q1656682 > mc: 16 > offset: 3333
```

FIGURE 4-12 – EXTRACTED ANNOTATIONS WITH TYPES

Finally, it should be mentioned that padding existing annotations with ontologically related concepts is not necessarily limited to *types*. Wikidata contains many other kinds of relationships[32], ranging from listing family members of celebrities to the type of material in a structure. However, for the scope of this thesis, the ontological exploration is limited to just types.

STEP 4 – COMBINE WITH PRE-TRAINED WORD2VEC VECTORS

The next step in the pipeline is to fetch Word2Vec vectors for the annotations extracted from DBpedia. This is achieved by joining the set of annotations with Google's pre-trained Word2Vec model discussed in Section 4.1.6. Using the pre-trained vectors is as easy as downloading the model[33], and decoding its binary format into a big map of $(Freebase, Vector)$ key-value-pairs. The map can then be used as an ordinary data structure.

The extraction process this far has already fetched Freebase IDs for all matching entities, thus, the actual pairing just involves adding Word2Vec vectors to the annotations as an extra attribute during runtime.

As mentioned briefly in Section 4.1.6, the Word2Vec vectors from the Google model are 1000-dimensional. However, unlike traditional BoW vectors, where every value represents a given *phrase*, the features of a Word2Vec vector are not as easy to grasp – they are

[32] *Wikidata Properties – https://www.wikidata.org/wiki/Wikidata:List_of_properties/all*

[33] *Word2Vec Freebase model – https://docs.google.com/file/d/0B7XkCwpI5KDYaDBDQm1tZGNDRHc*

simply inferred features from the learning process itself, with no inherent meaning beyond the values themselves. For illustrative purposes, Figure 4-13 illustrates what an actual Word2Vec vector for *Zalmay_Khalilzad* looks like. Although the actual values in the vector convey little intuitive sense, it is a fair assumption that 1000 values may represent a more information about *Zalmay_Khalilzad* than the word itself is capable of – presumably containing implicit information about both context and related entities.

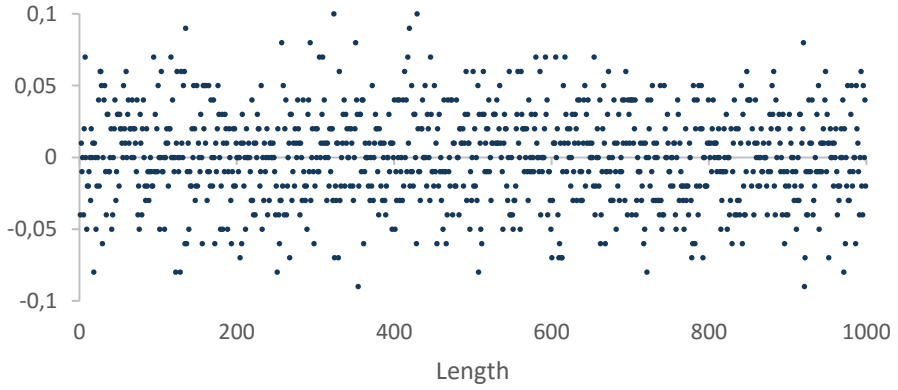


FIGURE 4-13 – WORD2VEC REPRESENTATION FOR ZALMAY_KHALILZAD

STEP 5 – FILTER AND SPLIT

A consequence of some of the preprocessing procedures in the preceding pipelining steps, is that many articles may end up having all of their categories or annotations filtered away. For this reason, the preprocessing pipeline is concluded with two filtering steps:

1. Filter annotations
    - Remove extracted DBpedia annotations with no Freebase match
    - Remove Freebase annotations with no Word2Vec match
2. Filter articles
    - Remove articles with no matching Media Topic category
    - Remove articles with no annotations

To emphasize a little on the removal of articles with no category; as explained in Step 2, the mapping from NYT descriptors to IPTC Media Topics is not perfect. Not every descriptor will have a match, and this leaves us with orphaned articles. As these articles provide no further information apart from the fact that they *do not have a category*, these articles are discarded.

For a visual feel of the filtering process and the dataset as a whole, Figure 4-14 shows the filtering results on the entire corpus. As the figure shows, the filtered dataset contains roughly 1.4 million news articles, and almost 22 million annotations, with about 213 thousand annotations being unique. Figure 4-15 illustrates the article breakdown per category, according to the *Broad* IPTC MediaTopic mapping explained in Step 2.

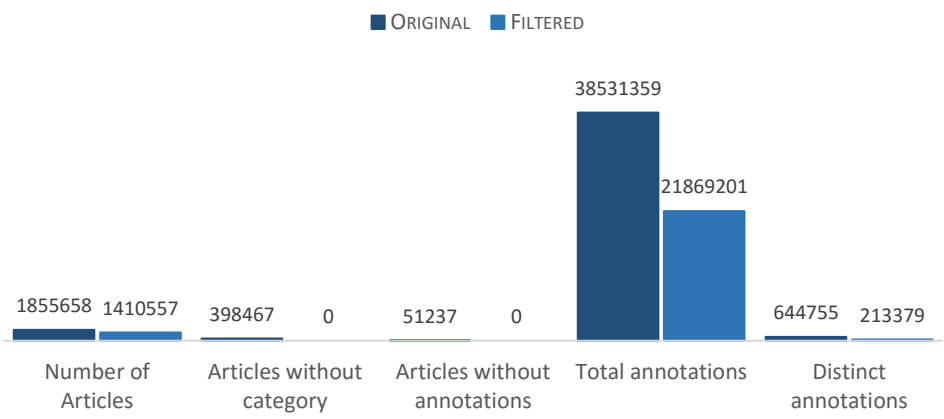


FIGURE 4-14 – CORPUS FILTERING RESULT

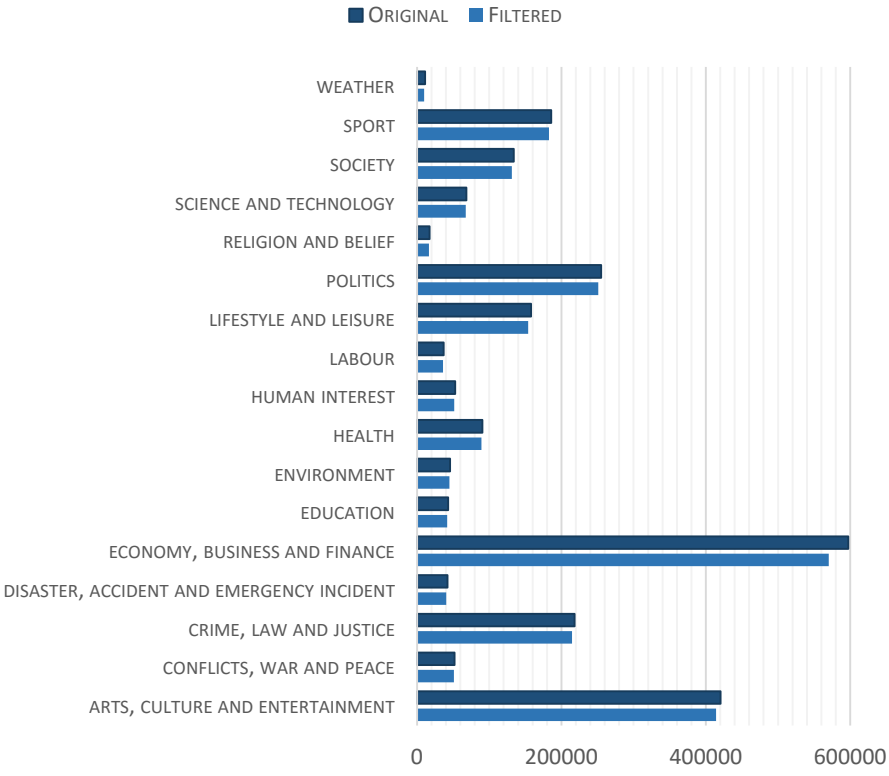


FIGURE 4-15 – ARTICLE DISTRIBUTION PER CATEGORY

After the cleanup phase has completed, the dataset is ordered chronologically by date, and split up into training, validation, and test sets. The reason for ordering the dataset by date is to avoid bias in the classifiers. Since the goal is to model news articles, which naturally reflects current events based on past, it makes sense to follow this scheme through the classification process as well. Exactly how the splitting is done depends on the experiment, but it generally follows the following metric:

$$60\ \%\ train, 20\ \%\ validation, 20\ \%\ test$$

The training set is used to *train* the models, while the validation set is used to evaluate, and help tune the hyperparameters during training. After training is complete, the finalized models are evaluated with the *test* set.

After the dataset is split up, the final step in the preprocessing pipeline is to calculate TF-IDF weights for all of the annotations. As covered in Section 2.2.2 TF-IDF is the *de facto* method of weighting features in a dataset.

NB: An important note is that the TF part of TF-IDF (i.e. term frequency in the entire corpus), is only computed based on the training set. This is done to avoid any bias in the training process. With other words: When calculating TF-IDF for annotations in the validation or test set, the TF from the training set is used.

4.2.2 DEEPER INSIGHT

Now that the dataset is preprocessed and split up accordingly, it is interesting to take a closer look at the data itself – both to understand the data a bit better before using it for training and as a sanity check to make sure the dataset still inhibits its desired properties.

The intended purpose of the annotated dataset is to have the extracted DBpedia entities convey the information present in the news articles. However, after all of the preprocessing done, it is not obvious which annotations actually represent the different categories. To get a better intuition of this, Table 4-4 has been included – which lists the most relevant annotations in terms of TF-IDF weight for each and every category.

Although some entries are less clear than others, one could argue that the bulk of them make sense. For instance, *ozone, greenhouse,* and *carbon dioxide* definitely seem relevant for *weather* articles. Among the results are also some less obvious, and perhaps amusing entries: Apparently *goldfish* and *caviar* applies to *human interest.* One thought-provoking remark, though, is that *salt* is assumed relevant by *four* categories, in the top-nine alone, which may suggest some overlap among them.

Some other interesting observations about the dataset is the average number of annotations for articles in each category, as seen in Figure 4-16. According to the plot, one should expect about half the amount of *information* from articles labeled *economy* as opposed to *religion and belief.* The average label cardinality [2.8.1] for the different

categories is displayed in Figure 4-17, which shows that most articles in the dataset have between 2 and 3 assigned categories.

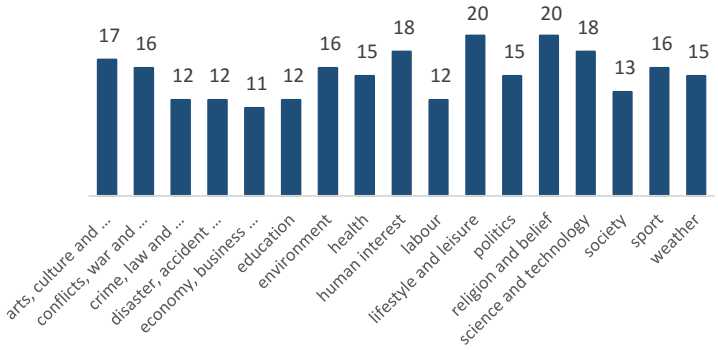


FIGURE 4-16 – AVERAGE NUMBER OF ANNOTATIONS PER IPTC CATEGORY

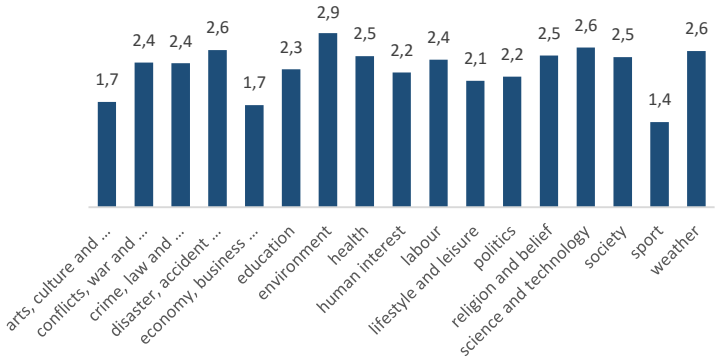


FIGURE 4-17 – LABEL CARDINALITY BREAKDOWN BY CATEGORY

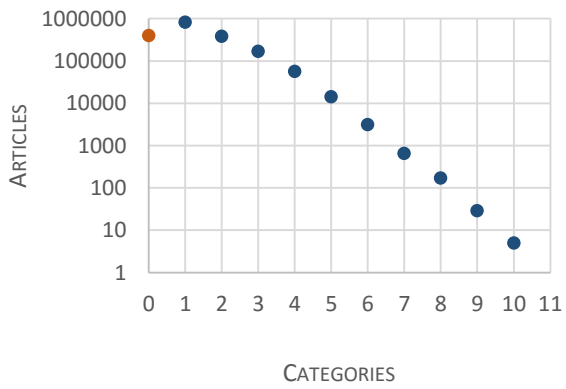


FIGURE 4-18 – LOGARITHMIC PLOT SHOWING MULTI-LABELNESS – THE NUMBER OF CATEGORIES PER ARTICLES

ARTS, CULTURE ...	CONFLICTS, WAR ...	CRIME, LAW ...	DISASTER, ACCIDENT ...	ECONOMY, BUSINESS ...
bilingual education	sonar	Gangi	cyclone	Isetan
confetti	Coverdale	Lubin	Valujet	Borden
National Geographic	Noriega	Hoffa	tritium	honeycomb
Univision	Mobutu	C.F.T.C	sonar	Univision
Goosebumps	Osama bin Laden	Governor	deer	easement
Genie	Kurds	Sergeant Major	boulevard	Peoplesoft
Lindbergh	Texaco	lysine	air bag	Oshkosh
pygmies	Chechnya	Petrocelli	tornado	C.F.T.C

EDUCATION	ENVIRONMENT	HEALTH	HUMAN INTEREST	LABOUR	LIFESTYLE ...
bilingual education	salt	chi	caviar	Workfare	gypsum
Ubinas	Georgian	dyslexia	liver	Reno	Grand Union
chancellor	Pinelands	euthanasia	deer	Greyhound	batik
Ledyard	asbestos	melanoma	PCB	Social Security	saddle
Head Start	Asbestos	varicose veins	Atlantic salmon	Lynn	plywood
Barnwell	Manville	bulimia	whaling	Saturn	muffler
Walden	Con Edison	autism	goldfish	workfare	salt
Asbestos	tuberculosis	Rogaine	fat	Governor	Saturn

POLITICS	RELIGION ...	SCIENCE AND TECHNOLOGY	SOCIETY	SPORT	WEATHER
Lubin	Kwanzaa	dyslexia	bilingual education	Ramsey	radar
Workfare	Koran	melanoma	Goosebumps	Nebraska	ozone
Hoffa	Rebbe	bulimia	Sergeant Major	Ledyard	carbon dioxide
impeachment	Barbie	autism	Nelson Mandela	Polonia	greenhouse
Schumer	Hasidic	Radon	Workfare	Spira	levee
Lockheed	Shaker	salt	Philippines	Schwinn	Kyoto
chiropractic	chocolate	American Red Cross	dyslexia	Sampras	salt
Social Security	Baptist	Head Start	euthanasia	Navratilova	river

TABLE 4-4 – MOST RELEVANT ANNOTATIONS IN TERMS OF TF-IDF FOR EACH CATEGORY

Furthermore, Figure 4-18 shows a logarithmic plot of the multi-labelness for the dataset – the distribution of number of categories assigned to the different articles in the corpus. This plot reveals that the vast majority of articles after the filtration have either one or two categories. The red dot plots the number of articles filtered away as a result of having no category after the initial filtering.

Finally, one particularly interesting set of relationships is shown in Table 4-5. This comprehensive table shows the correlational probability of *row* given *column*. For instance, the probability than an article that already has a *labour* category *also* has the *economy* label is 50 %. Looking through the table most of the relationships tend to make sense: A weather article is likely also to be about *disaster*, *economy* or *environment*. This is reassuring, as it implies that the IPTC Media Topic category mapping was successful.

	ARTS	CONFLICTS	CRIME	DISASTER	ECONOMY	EDUCATION	ENV	HEALTH	HUMAN	LABOUR	LIFESTYLE	POLITICS	RELIGION	SCIENCE	SOCIETY	SPORT	WEATHER
ARTS	-	19 %	18 %		12 %	23 %	14 %	16 %	24 %	15 %	16 %	18 %	28 %	22 %	26 %	5 %	8 %
CONFLICTS	2 %	-	5 %	3 %	2 %	1 %	2 %	1 %	1 %	2 %	1 %	10 %	4 %	2 %	2 %	0 %	1 %
CRIME	8 %	20 %	-	17 %	10 %	15 %	14 %	17 %	6 %	14 %	8 %	20 %	17 %	10 %	30 %	5 %	4 %
DISASTER	1 %	2 %	4 %	-	4 %	1 %	15 %	3 %	3 %	1 %	3 %	1 %	3 %	3 %	3 %	2 %	27 %
ECONOMY	19 %	22 %	33 %	56 %	-	25 %	49 %	31 %	21 %	50 %	27 %	35 %	16 %	35 %	25 %	6 %	40 %
EDUCATION	2 %	1 %	3 %	1 %	1 %	-	2 %	3 %	3 %	7 %	1 %	2 %	4 %	3 %	5 %	0 %	1 %
ENV	1 %	2 %	3 %	16 %	3 %	2 %	-	5 %	10 %	1 %	8 %	2 %	1 %	9 %	2 %	1 %	24 %
HEALTH	4 %	2 %	8 %	6 %	4 %	6 %	11 %	-	10 %	5 %	4 %	3 %	4 %	39 %	12 %	3 %	9 %
HUMAN	3 %	1 %	1 %	3 %	2 %	4 %	12 %	6 %	-	1 %	5 %	1 %	2 %	7 %	2 %	3 %	7 %
LABOUR	1 %	1 %	2 %	1 %	3 %	6 %	1 %	2 %	1 %	-	1 %	2 %	1 %	1 %	3 %	2 %	1 %
LIFESTYLE	6 %	4 %	6 %	11 %	6 %	5 %	28 %	8 %	16 %	6 %	-	4 %	36 %	9 %	6 %	11 %	12 %
POLITICS	11 %	48 %	24 %	8 %	13 %	13 %	13 %	8 %	4 %	18 %	6 %	-	15 %	8 %	23 %	1 %	5 %
RELIGION	1 %	1 %	1 %	1 %	0 %	1 %	0 %	1 %	0 %	0 %	3 %	1 %	-	1 %	2 %	0 %	1 %
SCIENCE	3 %	2 %	3 %	4 %	3 %	5 %	12 %	25 %	8 %	2 %	4 %	2 %	3 %	-	5 %	1 %	11 %
SOCIETY	8 %	5 %	21 %	9 %	5 %	17 %	6 %	18 %	4 %	10 %	5 %	13 %	17 %	12 %	-	2 %	3 %
SPORT	3 %	1 %	5 %	8 %	2 %	2 %	3 %	7 %	12 %	13 %	15 %	1 %	1 %	2 %	3 %	-	6 %
WEATHER	0 %	0 %	0 %	6 %	1 %	0 %	5 %	1 %	1 %	0 %	1 %	0 %	0 %	2 %	0 %	0 %	-

TABLE 4-5 – PROBABILITY OF ARTICLE IN CATEGORY ROW ALSO BEING CATEGORY COLUMN

4.2.3 FINALIZED MODEL

After the preprocessing pipeline is completed, the final model becomes rather slim in terms of original size – maintaining only the relevant attributes (with the exception of the *headline* attribute used for debugging purposes). The final article model is described in Table 4-6, while the annotation model is available in Table 4-7.

NAME	TYPE	DESCRIPTION
ID	String	Unique identifier
Headline	String	Article headline
IPTC	List	Correct IPTC Media Topics category
Annotations	Annotation	Relevant phrases

TABLE 4-6 – FINALIZED ARTICLE MODEL AFTER PREPROCESSING

NAME	TYPE	DESCRIPTION
ID	String	Unique identifier
Index	Integer	Per-article unique identifier
Phrase	String	Freebase entity name
Mention Count	Integer	Number of mentions within a single article
DBpedia ID	String	Unique identifier from DBpedia
Freebase ID	String	Freebase ID, /m/<string>
Wikidata ID	String	Wikidata ID, Q<number>
TF-IDF	Double	Term frequency, inverse documents frequency

TABLE 4-7 – FINALIZED ANNOTATION MODEL AFTER PREPROCESSING

5 METHODS

One of the major goals of this thesis is to investigate and evaluate how simple traditional machine learning compares to the more sophisticated neural approaches. For this reason, careful thought has to go into the process of picking the appropriate methods suitable for providing insight on this matter. One concern when dealing with this is to make sure the methods are compared on fair ground – or with other words: That the comparison evaluates the *methods* and not the surrounding circumstances. The purpose of the following chapter is to explain the "which and why" behind the choice of classifiers, and their configurations. Practicalities surrounding the training phase, and how the preprocessed dataset from Chapter 4 was tailored to fit the input format for each classifier, is also included in this chapter.

5.1 TESTING ENVIRONMENT

The testing environment used for this thesis consist of a custom classification framework written in Scala for Apache Spark[34]. The framework uses tools from Spark's MLlib library (v1.6.1) for the Naïve Bayes implementation, and deeplearning4j (v0.4-rc3.10) for configuring and training of ANNs. Furthermore, the developed framework is open source and available on GitHub[35]. To cope with the computational demands, training was done on a series of Intel Xeon E5 32 core 2.6ghz CPUs.

5.2 CLASSIFIERS

In order to keep the multi-label classification task as simple as possible, the news article categorization is modeled as an n-binary classification problem. This choice is also motivated by the promising related work using n-binary ensembles, discussed in Section 3.1. As explained in Section 4.1.2 this means that no less than 17 binary classifiers are required for each of the chosen approaches – one for each of the top-level IPTC Media Topic categories.

There are three classifiers used in this project:

(i) *Multinomial Naïve Bayes (NB) – [2.4.1]*

(ii) *Feedforward Multilayer Perceptron network (FFN) – [2.5.1]*

(iii) *Long Short-Term Memory Recurrent neural network (LSTM) – [2.5.6]*

[34] *Spark – http://spark.apache.org/*

[35] *Corpus Project – https://github.com/Habitats/corpus*

The motivation behind this choice of classifiers is to represent three different levels of sophistication. Summarized, NB acts as the baseline for the traditional methods. SVM [2.4.2] and k-NN [2.4.3] were also explored as alternatives to the traditional method, but NB was ultimately chosen because of its simplicity and low computational requirements. For the more complex methods, the FFN represents the baseline for the *artificial neural networks* (ANN), while the LSTM represent a more sophisticated alternative. As a quick recap: LSTM are networks capable of modeling ordered sequences, which makes them particularly interesting for modeling text.

The following section provides a breakdown of how each of the classifiers was configured and tuned to fit the news article classification problem.

5.3 FEATURE SELECTION

Once the data preprocessing pipeline demonstrated in Section 4.2 is completed, the dataset consists of a set of articles, each with one or more annotations attached to it. In order to make the articles and annotations compatible with a classification model, the dataset has to be translated into a format that the classification algorithm can understand. As discussed in Section 2.2, the representations that were chosen in this thesis are either a *bag-of-words/TF-IDF* approach, or through *word embeddings*.

5.3.1 BAG OF WORDS WITH TF-IDF

The first representation involves creating a standard BoW vector of TF-IDF weights for every article. Figure 5-1 illustrates how an example of how a small corpus may look like after being translated into TF-IDF vectors. *United States* and *Iraq* have column values of zero, implying they were present in all of the articles, and hence not important.

Freebase	/m/01fqm	/m/03shp	/m/06pq6	/m/09c7w0	/m/0d05q4
WikiData	Q1530	Q794	Q9585	Q30	Q796
Phrase	Baghdad	Iran	Shia Islam	United States	Iraq
Article 1	0,000	0,000	0,000	0,000	0,000
Article 2	0,000	0,330	0,045	0,000	0,000
Article 3	0,378	0,000	0,000	0,000	0,000
Article 4	0,000	0,231	0,162	0,000	0,000

FIGURE 5-1 – A FINAL MATRIX REPRESENTATION OF A SMALL PREPROCESSED CORPUS

Using the BoW representation together with the NB and FFN is as simple as mapping the vector directly to the input. However, as the preprocessed corpus contain no less than 213 379 unique annotations [Figure 4-14], some additional filtering is carried out to make the method tractable in terms of computation time – which in terms of the chosen methods is proportional to the size of the input vector.

Therefore, annotations with a term frequency less than 100 were ignored. This reduced the number of annotations to around 25 000, without affecting classification performance in any noticeable way. Other dimensionality reduction techniques like PCA and *feature selection,* as explained in Section 2.6, were attempted, but not used because of the lack

of suitable implementations in MLlib. PCA in particular turned out to be very slow to compute.

Furthermore, using the BoW representation together with the LSTM was shown highly impractical due to the computational requirements, and was for this reason omitted. That is, the BoW representation was only used with FFN and NB.

5.3.2 WORD EMBEDDINGS WITH WORD2VEC

Utilizing the embedded Word2Vec representation is a little less straightforward than the BoW approach. As a recap from Section 4.2: With the chosen Word2Vec implementation, each article has a set of n_a 1000-dimensional vectors, where n_a is the number of annotations in article a. This means that every article can be seen as a 2-dimensional $n_a * 1000$ *matrix*. However, as soon to be uncovered, simple vector representation is also possible.

VECTORIZED REPRESENTATION

The matrix representation becomes an issue when working with NB or FFN, which expects vectorized inputs. This means that the article has to be translated yet another time in order to fit as input for these classifiers.

As all of the vectors have 1000 dimensions, there are several ways the information in the vectors can be accumulated into a lower dimensionality representation. For instance, by taking the average, adding them together, or concatenating [114]. The method that was proven the most useful for this thesis, consisted of multiplying the vectors with their corresponding annotations' TF-IDF weight, and *then* adding them together – ultimately creating a *document vector* from the set of Word2Vec phrase vectors. Not only did this simplify the input, but it also created a cumulative footprint for the article in light of the TF-IDF weights from its corresponding annotations.

By using the document vector approach, the representation was reduced to the same format as the BoW vectors, only with 1000 dimensions instead of a dimension equal to the number of distinct annotations.

MATRIX REPRESENTATION

While the NB and FFN methods are limited to vectorized inputs, the LSTM is capable of utilizing the full matrix representation. As elaborated in Section 2.5.6, an LSTM can model *time,* which for the news article classification problem can be interpreted as the internal *order* of the annotations within an article – similar to how a human would process a text from left to right.

In practice, this means every row in the document matrix represents a single *time-step* for the LSTM, a time-step which is reset after every article. Training is done the same way as the other methods, only with matrices as input instead of vectors.

Incorporating TF-IDF weights into the ordered Word2Vec representation used with the LSTM did not show much effect. For this reason, the TF-IDF weights were not used with the LSTM.

5.4 EVALUATION METRICS

To fully explore the properties of the data, numerous evaluation metrics are investigated. For the label-based metrics, this includes both the macro- and micro-averaged *precision, recall, accuracy* and *F-score*. Example-based metrics that are used are *Hamming loss* (H-Loss) and *Subset accuracy* (Sub-Acc). In addition, a modified Subset-accuracy metric, Subset-one accuracy is used, which is the fraction of articles with *at least one* correctly predicted category. *Layer Cardinality* (LCard) is also investigated, sometimes normalized in terms of the *true* label cardinality. That is, a normalized LCard of 2.5 would mean the classifier predicted 2.5 times more categories for the article than what the article actually has.

As discussed in Section 3.4.3, the most commonly used evaluation metric when analyzing news articles is the macro-averaged F-score. For this reason, the macro-averaged F-score metric used for the final assessment. All further mentions of F-score assume macro-average unless explicitly stated otherwise.

As a refresher from Section 2.7: The macro-averaged strategy first calculates the metrics individually for each label, and *then* takes the average – making it the sensible choice for news data, where the label classes differ a lot in size, but are equal in importance.

5.5 HYPERPARAMETER TUNING

A simple learning problem can be viewed as a function taking some training data and producing one or more outputs. However, in reality, this process usually also involves tuning a set of hyperparameters. Said best by Yoshua Bengio, one of the fathers of modern A.I., hyperparameters are "annoying knobs to be adjusted" [115]. Or put another way; hyperparameters are the parameters of the learning algorithm itself.

Although simple methods like Naïve Bayes have very few knobs, entering the realm of neural networks, the complexity, and level of sophistication skyrockets. Even for FFNs, which are arguably the simplest ANNs capable of performing classification, there are already a wide variety of hyperparameters to tune. Tuning LSTMs is no different – if anything, it is even more complicated.

The hyperparameters deemed relevant for this thesis are primarily those covered throughout Section 2.5 – ranging from the choice of learning rate, and activation functions, to choosing the number of neurons in a layer, or even the number of layers in the network itself.

Needless to say, choosing suitable hyperparameters when starting from scratch with a previously unresolved classification problem is far from trivial. In fact, one of the most

prominent hyperparameter optimization techniques, *Random Search,* boils down to just being a matter of educated guessing [116]. In short Random Search involves figuring out a suitable range for every hyperparameter, often initiated by looking at related problems or following rules of thumb, and then just trying random permutations of parameters *until it works.*

Random Search represents more or less how the hyperparameter search was conducted for this thesis. "More or less", as it was done with simplicity in mind: Choose the simplest kind of network, and add bells and whistles until it either stops improving or becomes too complicated to work with. The following sections present the most promising configurations.

5.5.1 NAÏVE BAYES

As previously discussed, NB is an incredibly simple classifier, and requires little to no tuning in order to function. In fact, the only noteworthy hyperparameter is to decide whether it should be a multinomial or Bernoulli classifier. Multinominal means that it will predict discrete data, as opposed to the Bernoulli variant, which is probabilistic. The NB classifier for this thesis was chosen to be binary, which is a special case of the multinomial classifier, with only two classes – that is, it outputs a 1 if it thinks a certain article is in a given category, and 0 otherwise.

5.5.2 FEEDFORWARD NETWORK

The relevant hyperparameters and their configuration for the FFN are summarized in Table 5-1. One thing to notice is that there are two outputs through a Softmax function. This means that the network outputs a probability distribution for the binary choice $[yes, no]$. For instance, $[0.55, 0.02]$ would means the model would output a "yes".

Although there are no right or wrong regarding the number of nodes in a layer, the topology for the FFN W2V network was guided by a couple of rules of thumb proposed by Jeff Heaton [117], which states that:

(i) *The number of hidden neurons should be between the size of the input layer and the size of the output layer.*

(ii) *The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.*

(iii) *The number of hidden neurons should be less than twice the size of the input layer.*

Another important detail is that the topology of the network varies based on the representation of the input format. The original idea for the FFN BoW topology was to prepend another layer in front of the 1000 input FFN W2V network, with as many inputs as the BoW feature space required. This design was motivated by an assumption that this would make it easier to compare the two methods, as the topologies would be otherwise identical. However, the resulting network turned out to be extremely slow to

train, due to the high number of input nodes. For this reason, the FFN BoW network had to be kept very simple in order to remain a tractable solution in terms of computation time.

Other than that, most of the hyperparameters were found through trial & error as explained in Section 5.5.

It should be noted that Table 5-1 only shows hyperparameters for techniques that were actually used, and thus omits hyperparameters that had little to no effect on the training process in this particular scenario. Among the techniques that were tried, but yielded little impact, are *dropout* and *input scaling*.

Furthermore, some hyperparameters were *manually* dynamic – like the learning rate. What this means is that the networks were first trained using a higher learning rate, then saved and trained again with a lower learning rate. This procedure did to a certain degree mimic an adaptive learning rate. Though, the method ended up being a little cumbersome because of limitations in deeplearning4j.

HYPERPARAMETER	VALUE
OPTIMIZATION ALGORITHM	Stochastic Gradient Descent with Backpropagations. Mini-batch size of 250
UPDATER	RMSProp
WEIGHT INITIALIZATION	Xavier
ACTIVATION INPUT	Hyperbolic tangent
ACTIVATION OUTPUT	Softmax
COST FUNCTION	Cross Entropy
TOPOLOGY WORD2VEC	1000 inputs, 700 in the first hidden layer, 500 in the second and 2 outputs
TOPOLOGY BOW	25000 inputs, 1000 in the first hidden layer and 2 outputs
LEARNING RATE	Initially 0.5, then decreased gradually until 0.05

TABLE 5-1 – HYPERPARAMETER CONFIGURATION FOR FFN

5.5.3 LONG SHORT-TERM MEMORY NETWORK

As shown in Table 5-2, the hyperparameter configuration of the LSTM is very similar to the FFN in Table 5-1. The most noteworthy change is the topology. Although the LSTM units are a lot more complex than the simple nodes in the FFN [Figure 2-18], 100 hidden nodes are still very few compared to the 1000 nodes in the first layer. However, due to LSTMs incredibly demanding computational requirements, 100 was the highest number of nodes the available resources allowed for.

Other than that, the hyperparameters for the LSTM were chosen much the same was as those of the FFN, using simplicity and Heaton's three rules of thumb as a guideline.

HYPERPARAMETER	VALUE
OPTIMIZATION ALGORITHM	Stochastic Gradient Descent with Backpropagations. Mini-batch size of 50. Gradient clipping applied with a threshold of 1.0
UPDATER	RMSProp
WEIGHT INITIALIZATION	Xavier
REGULARIZATION	L2 regularization with a coefficient of $1 * 10^{-5}$
ACTIVATION INPUT	Softsign
ACTIVATION OUTPUT	Softmax
COST FUNCTION	Cross Entropy
TOPOLOGY WORD2VEC	1000 inputs, 100 hidden LSTM units, and 2 outputs
LEARNING RATE	Initially 0.5, then decreased gradually until 0.005

TABLE 5-2 – HYPERPARAMETER CONFIGURATION FOR LSTM

5.6 TRAINING TIME

The previous sections mention that the chosen topologies for the FFN and LSTM were heavily constrained by the high computational requirements involved when working with ANNs. To put some numbers on these constraints: A *single* run through the full dataset, using an LSTM with only 10 hidden nodes, was initially estimated to take more than 45 days, using on an Intel Xeon E5 machine.

However, after spending countless hours working closely with the developers of *deeplearning4j,* implementing a variety of clever caching techniques, tuning garbage collectors, and adding native bindings for the mathematical components, the training time was finally cut down to less than 3 days for a 100 node LSTM. Yet, as evident from Figure 5-2, the training time for the bigger ANNs do not even come close to the simple nature of Naïve Bayes.

The motivation behind mentioning this, is to shed some light on the many unforeseen issues faced when working with something as complex as ANNs. Another side effect of the long training times is that it inhibits the number of different hyperparameter configurations possible to explore with Random Search. With an epoch time (a single run through the dataset) of two and a half days for the LSTM, time quickly adds up.

It is also worth mentioning that *prediction* time – the time it takes for the model to evaluate a single example – was observed to increase proportionally to the network size. In fact, testing 200 000 examples on a finished LSTM model takes around five hours, while the NB completes in a matter of minutes.

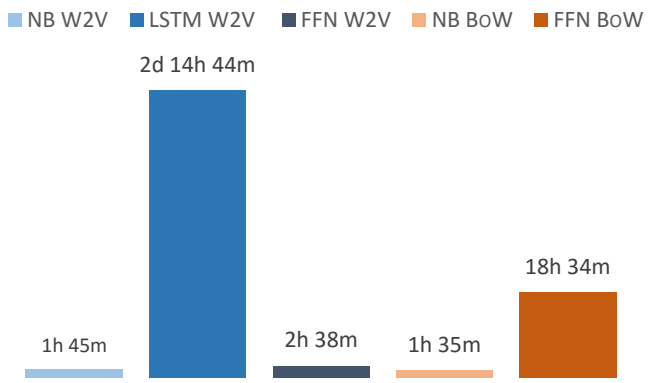


FIGURE 5-2 – TRAINING TIME FOR A SINGLE EPOCH ON THE FULL DATASET

6 EXPERIMENTS

During the introductory chapter, a set of research questions were defined [1.4]. This analytical chapter attempts to address these questions, by conducting a set of experiments. In total, seven experiments have been carried out, each covering different aspects of the research goals. As the research questions and experiments are somewhat intertwined, a mapping showing the experiments' main focal point, is presented in Table 6-1. RQ7 is not included as it is addressed throughout Chapter 5 and 6 as a whole.

In general, the experiments can be split into two groups. The first group is concerned with how the dataset may be preprocessed to enhance training. This involves decisions regarding the number of annotations necessary, and whether or not to incorporate ontologically related concepts into the training phase. The second group of experiments revolves around analyzing and comparing the different methods. For instance, by exploring how article length and moving through time affects the quality of the classification.

NB: For the sake of keeping the evaluation as clean as possible, the *results* section for each experiment only focus on delving into what is relevant with respect to its related research question.

	RQ1	RQ2	RQ3	RQ4	RQ5	RQ6
6.1 IMPORTANCE OF CONFIDENCE			X			
6.2 ONTOLOGICAL					X	
6.3 ARTICLE LENGTH			X			X
6.4 PERFORMANCE OVER TIME				X		
6.5 WORD EMBEDDINGS VERSUS BAG OF WORDS		X				
6.6 NAÏVE BAYES VERSUS DEEP LEARNING	X					
6.7 WORDS VERSUS ANNOTATIONS						X

TABLE 6-1 – OVERVIEW OVER EACH EXPERIMENTS MAIN FOCUS POINT

6.1 IMPORTANCE OF CONFIDENCE

As explained in Section 4.2.1, the feature extraction from the articles is done through DBpedia Spotlight. The default confidence level for this extraction was set to 50 %. This first experiment attempts to justify this choice by investigating how different confidence levels affect the classification process.

It should be taken into consideration that this experiment merely serves as a means to an end in order to build a viable foundation for the remaining experiments. For this reason, the evaluation does not dig deeper than what is necessary to advocate a sensible confidence level.

PREPARATION

Figure 4-9 already gave a good indication of what to expect from the different confidence levels, by showing that a lower confidence yields a higher number of annotations. To get a better picture of exactly *how* many more, Figure 6-1 shows the average number of annotations per article for a random sample of a sample of 45 000 articles. An implication following the massive number of annotations for the 25 % level, is that the task of testing the model on the full dataset becomes infeasible. The number of annotations per article, 88 on average, simply becomes too large to be practical. For this reason, this experiment was carried out with only a small portion of the full dataset, totaling 45 000 articles. The default split from Section 4.2.1 was used.

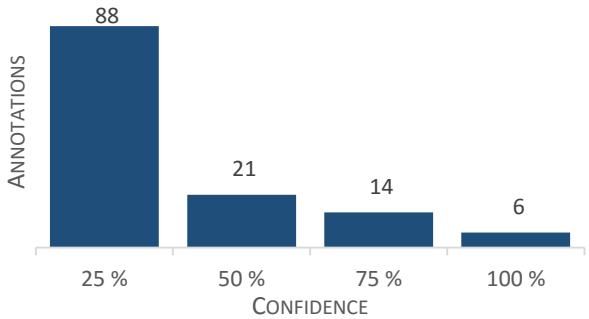


FIGURE 6-1 – AVERAGE NUMBER OF FOR EACH CONFIDENCE LEVEL ON A SAMPLE DATASET

RESULTS

As for the results, Figure 6-2 shows the macro-averaged scores for the different confidence levels. Looking at the F-score reveals some mixed results: The FFNs seems to benefit more from a higher number of uncertain annotations, while the LSTM and NB tend to prefer a confidence around 50 %.

Either way, a confidence level around 50 % was found to be a sound choice: It gave all of the 5 methods enough to work with, while at the same time keeping the problem domain within tractable bounds. However, because of the limitation on dataset size, the fine-grained preferences remain inconclusive.

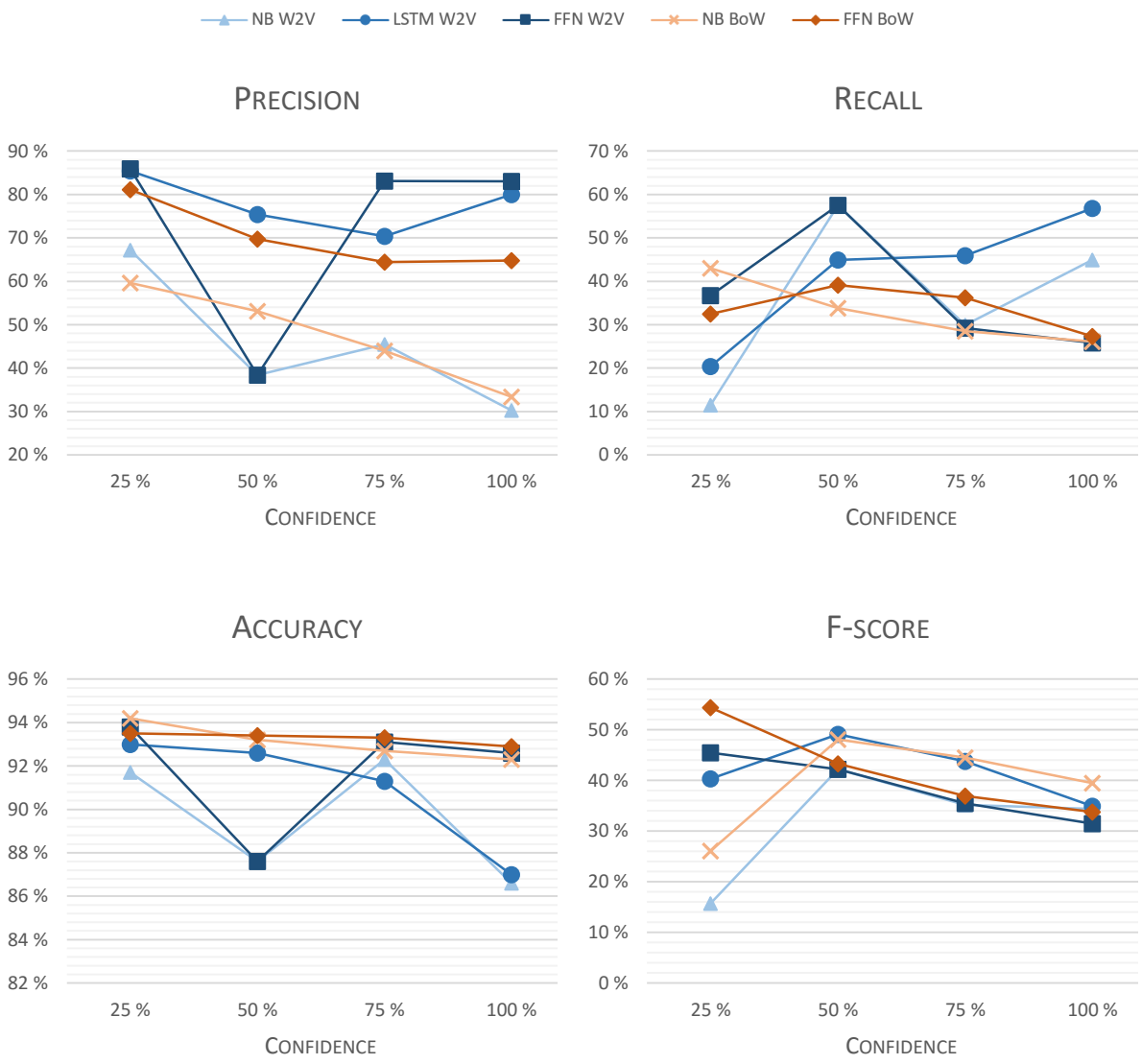


FIGURE 6-2 – IMPACT OF DIFFERENT CONFIDENCE LEVELS

6.2 ONTOLOGICAL EXTRACTION

The core motivator for this thesis is to explore language-independent news classification through feature extraction of ontological annotations. This experiment is devoted to delving deeper into the implications of doing so. The experiment also includes an assessment of the effects of incorporating the optional ontologically related supertypes from Section 4.2.1 into the classification process.

PREPARATION

For this experiment, the entire corpus was used. From this, two identical dataset splits were created, according to the standard splitting procedure [4.2.1]. One of the datasets was then appended with supertype annotations, whereas the other one remained as is.

Figure 6-3 illustrates how the number of annotations for each category was affected by adding types. An interesting observation is that padding with types had about the same effect on all of the categories, adding an average of 4 extra type annotations for every category.

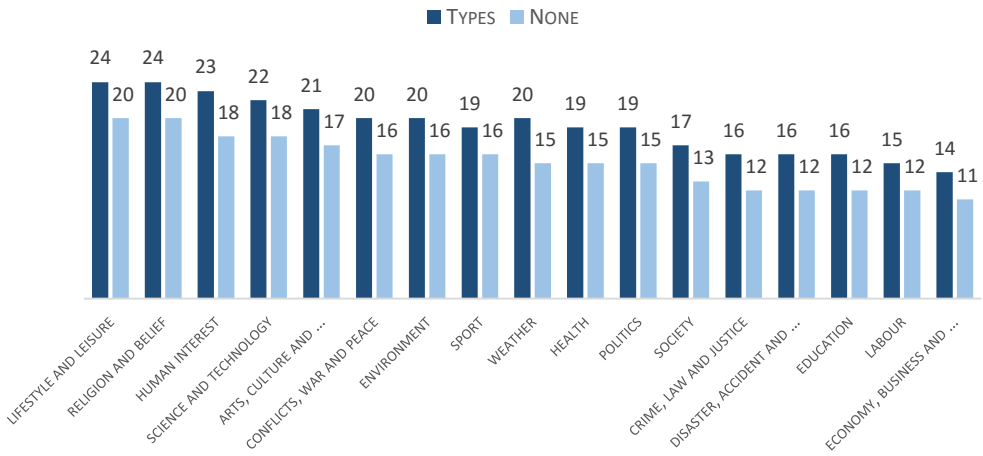


FIGURE 6-3 – AVERAGE NUMBER OF ANNOTATIONS PER CATEGORY WITH AND WITHOUT SUPERTYPES

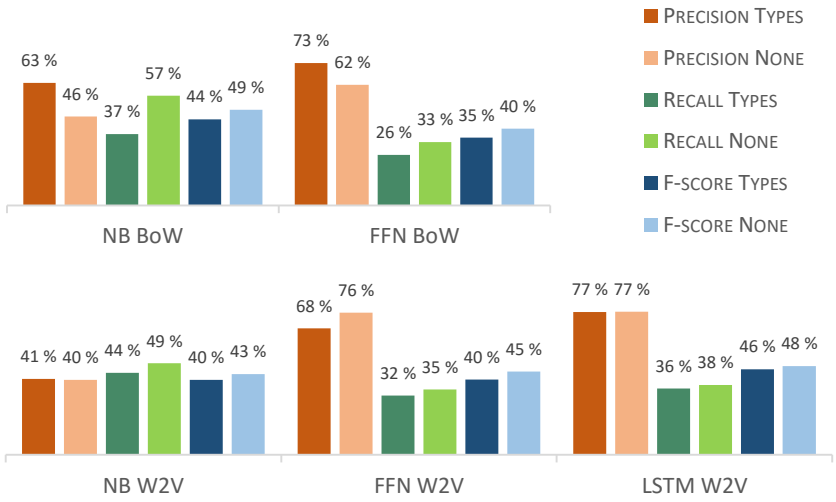


FIGURE 6-4 – EFFECTS OF INCLUDING ONTOLOGICAL TYPES. TYPES IN DARK COLORS, NONE IN LIGHT

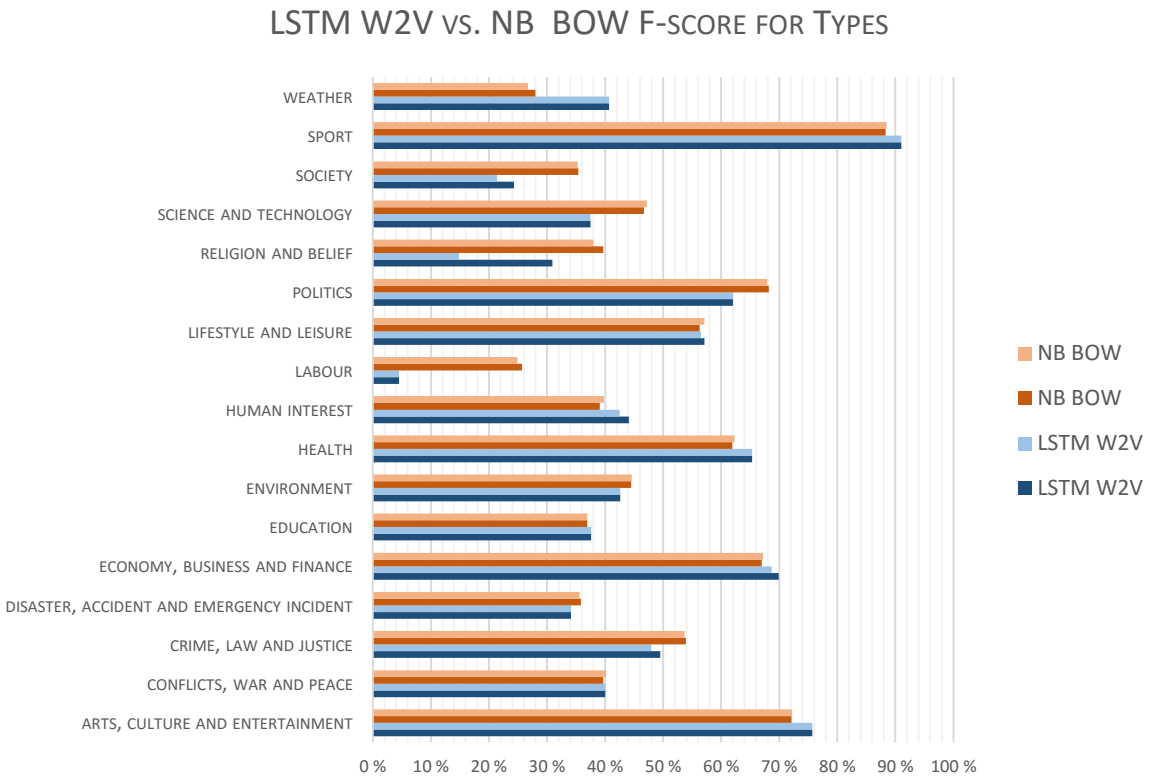


FIGURE 6-5 – LSTM W2V VS. NB BOW CATEGORICAL BREAKDOWN OF F-SCORE FOR TYPES

RESULTS

The macro-averaged F-scores for this experiment are shown in Figure 6-4, where the dark colored bars represent the scores with type inclusion, and the light colored bars without.

In regards to supertypes, the overall direction of the results is quite conclusive. For all of the five methods tested, the incorporation of types results in a slight drop in F-score. By closer examination, the results reveal an interesting meta-relationship: For both of the BoW-based methods the recall decreases while the precision *increases*, when adding types. That is, it swaps some false positives for false negatives. Exchanging false positives for false negatives effectively means reducing the label cardinality, as the classifier would refrain from misclassifying articles at the price of also *missing* some articles.

Although the accumulated F-values for every method decreased by introducing types, there is one particularly interesting observation to be made from the categorical breakdown from Figure 6-5. What the chart uncovers is that the decay is not uniform across categories. Looking at *religion and belief* the F-score is nearly cut in half for the LSTM W2V. Furthermore, the results show that the performance varies immensely between the different categories, even though the exact same method is used.

Intuitively, one may believe that incorporating types would only add *redundant* information, and not negatively affect the existing information. In addition, considering that the performance across the categories – with some exceptions – is relatively consistent between methods, the inconsistency between categories may originate from the dataset itself.

To investigate this hypothesis, remember the category distribution for the dataset in Figure 4-15. This shows that the distribution of categories is *extremely* skewed – or imbalanced. That is, the dataset contains a lot more info about certain categories than others. The effects relating this imbalance and the F-score is plotted in Figure 6-6. The plot shows the fraction of articles having a certain category with respect to the entire dataset, plotted against the F-score for the NB BoW classifier.

To emphasize: The imbalance factor in Figure 6-6 is the true/false rate. For instance, if a fourth of a dataset's articles are labeled *sport* the imbalance factor for *sport* would be 25%.

The imbalance correlation plot uncovers a clear relationship between the imbalance of a category and the resulting F-score for most categories: A more balanced category generally yields higher F-scores. However, there is one significant exception, namely *sport,* shown in red. *Sport* achieves the highest F-score while only inheriting an imbalance rate of 12 %. Thus, the imbalance rate cannot be the only side-effect affecting the F-score.

As it turns out, there is another correlation to be revealed by looking deeper into the properties of the annotations themselves. This relationship is related to the fraction of unique annotations for a given category, and is plotted in Figure 6-7. What the correlation plot shows, is that the fraction of unique annotations is inversely proportional to the F-score: Lower fraction of unique annotations yield higher F-scores. This observation is

sound in the sense that a greater number of unique annotations results in lower statistical significance for the features themselves. An even closer look actually hints towards a quadratic decay in F-score when increasing the number of distinct annotations.

Although only the NB BoW F-score was used for this detailed inspection, the same relationships were present for all of the other classifiers.

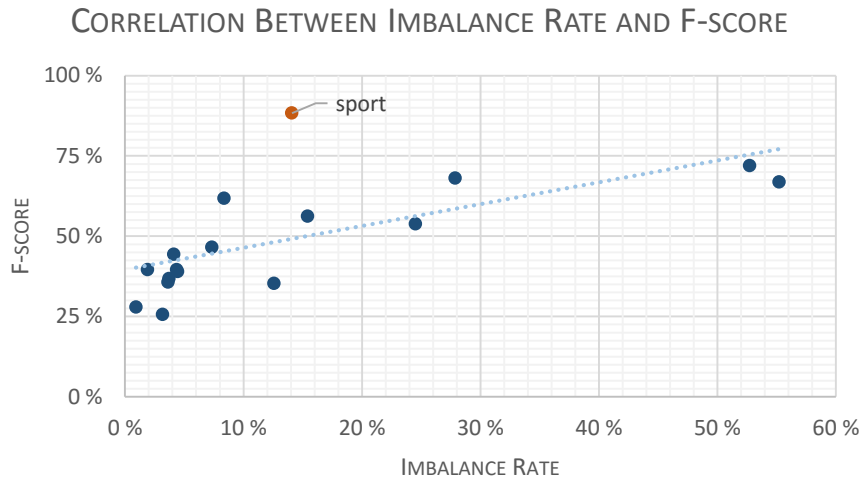


FIGURE 6-6 – CORRELATION BETWEEN IMBALANCE RATE AND F-SCORE FOR NB BOW

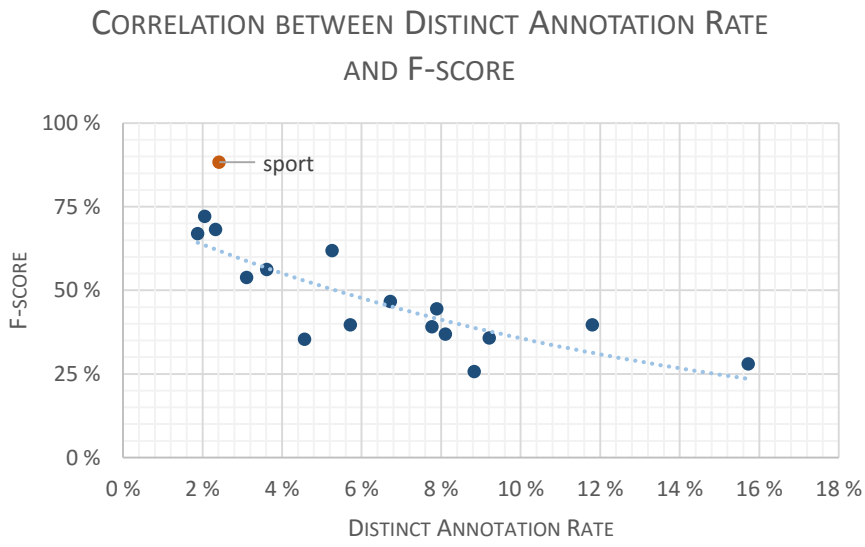


FIGURE 6-7 – CORRELATION BETWEEN DISTINCT ANNOTATIONS AND FOR NB BOW

6.3 ARTICLE LENGTH

Like most other types of media, news articles come in all shapes and sizes. It is a fair assumption that these properties may affect the classification quality. The following experiment attempts to uncover exactly how much of an impact this is – more specifically, how the performance of the classifier correlates with the number of annotations and article length.

PREPARATION

The only information available to the classifiers in these experiments are the annotations, and not the actual article text. For this reason, in order to say something about article *length*, there is a requirement for a correlation between article length and the number of annotations. Based on Figure 6-8, this correlation is assumed to exist.

In order to evaluate the models on different article lengths, the test set was split into ten equally sized buckets – each corresponding to a *length group* as shown in Figure 6-9. The models were then trained using the standard split defined in Section 4.2.1.

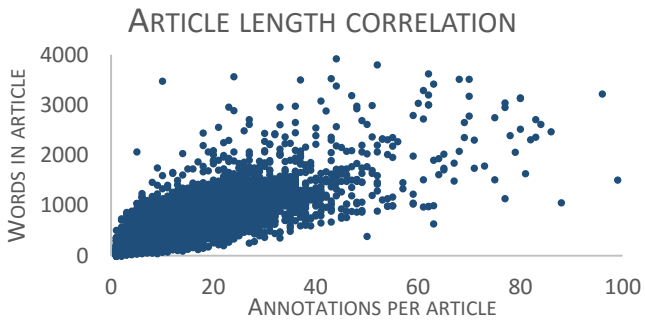
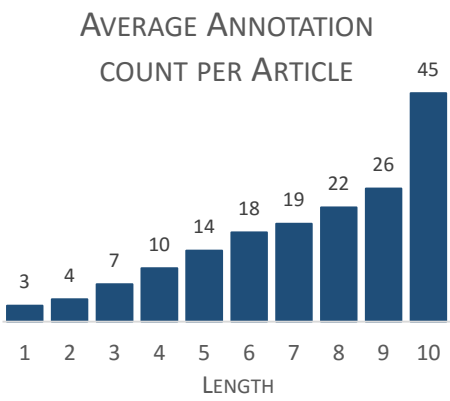


FIGURE 6-8 – CORRELATION BETWEEN NUMBER OF ANNOTATIONS AND ARTICLE LENGTH



GROUP	WORDS	ARTICLES
TRAIN	0 – 45355	846334
TEST 1	0 – 136	28212
TEST 2	137 – 191	28212
TEST 3	192 – 304	28212
TEST 4	305 – 500	28212
TEST 5	501 – 675	28212
TEST 6	676 – 810	28212
TEST 7	811 – 952	28212
TEST 8	953 – 1129	28212
TEST 9	1130 – 1441	28212
TEST 10	1421 – 45355	28212

FIGURE 6-9 – LENGTH BUCKETS

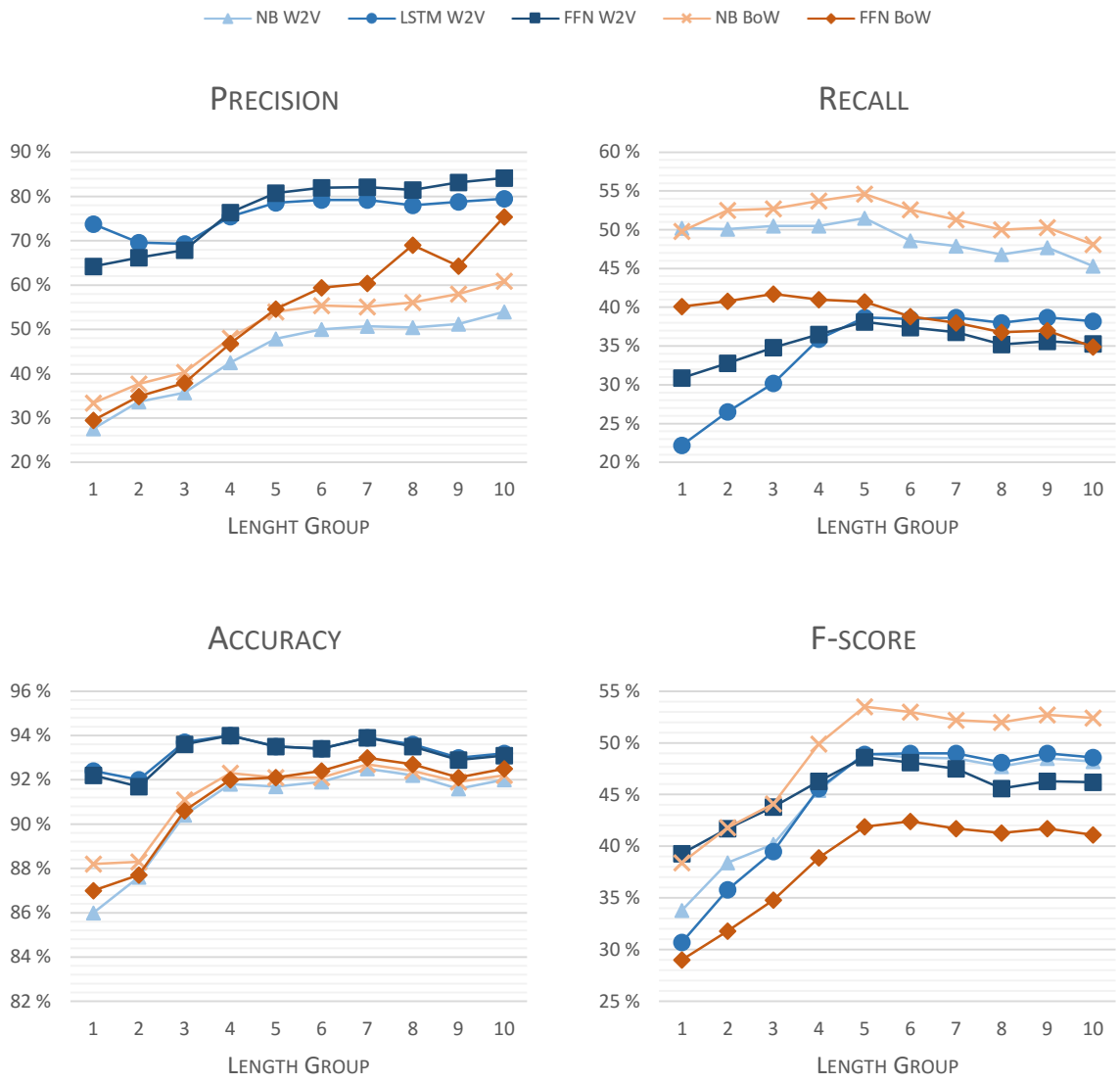


FIGURE 6-10 – CLASSIFICATION PERFORMANCE WITH RESPECT TO ARTICLE LENGTH

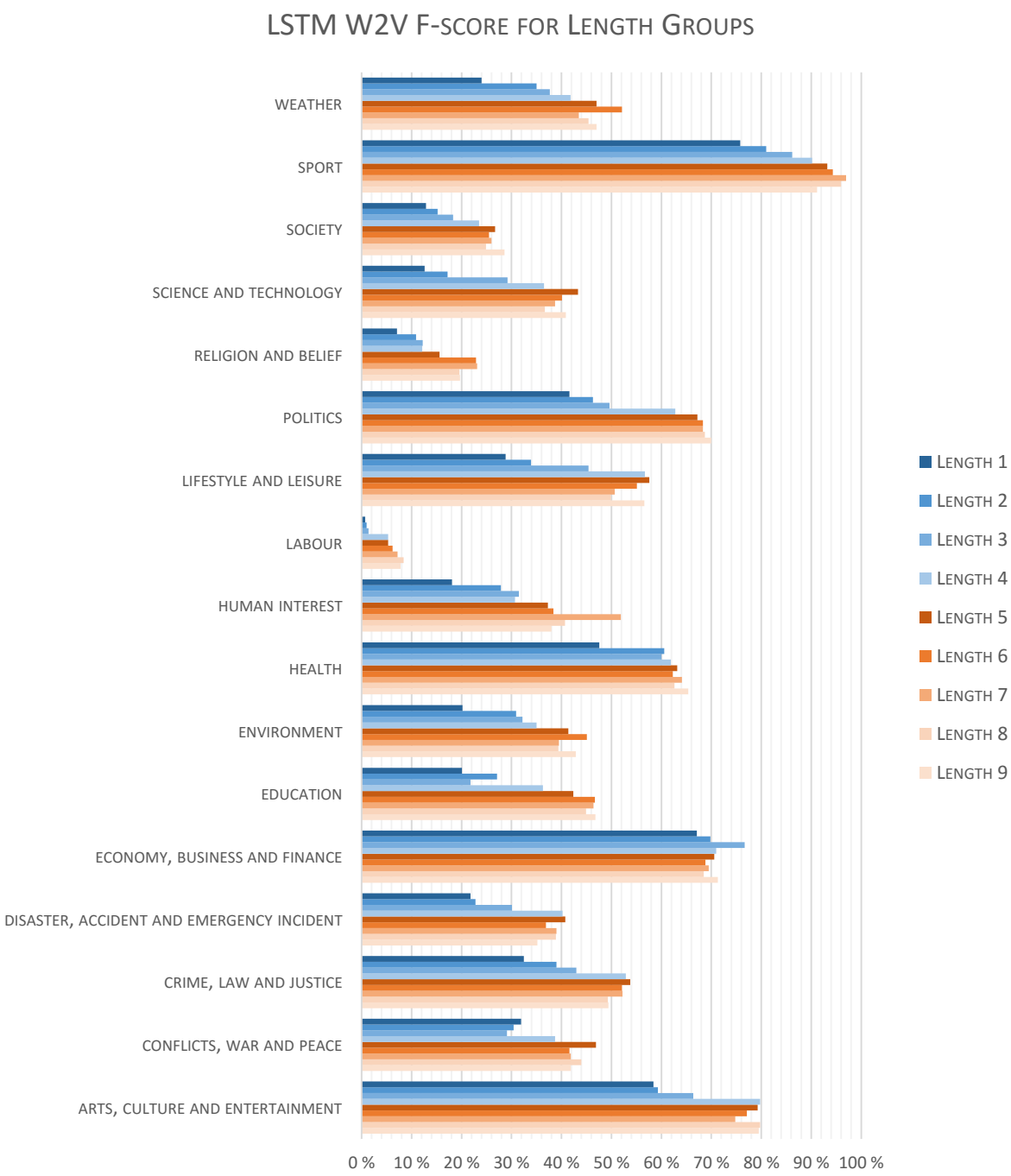


FIGURE 6-11 – PROGRESSIVE BREAKDOWN OF LSTM F-SCORE FOR LENGTHS, BLUE BELOW THE EDGE, RED ABOVE

RESULTS

Figure 6-10 captures the initial results from this experiment. Looking at the precision graph, it is possible to spot a clear increasing trend towards length group 5 – which translates into roughly 600 words. As all of the groups are of similar size, having the peak at around 5 implies that about half of the articles are equal or greater in length. The trend in precision translates naturally into the F-score and accuracy, as the increase in precision outweighs the subtle decrease in recall.

To better understand what is causing this rather significant edge in the F-score at group 5, a deeper dive into the categorical breakdown is required. This breakdown is depicted in Figure 6-11, with *blue* bars showing F-scores below the observed edge, and *red* bars representing the F-score above the edge. Looking at the specific categories, there seems to be a consensus regarding the correlation between longer articles and F-score. It is especially clear when looking at the categories s*port, politics* and *weather,* whose F-score's progression closely resembles the macro-averaged F-score development in Figure 6-10.

However, there are also exceptions to this relationship, *human interest* being the most significant one. For this category, transitioning into length group 6 actually *decreases* the F-score by more than 10 %. The reason for this abrupt behavior was first thought to be related to a skewed distribution of articles among the different length groups. A closer investigation found this not to be the case, as that distribution turned out to be almost uniform. Further assumptions remain speculative: It might just be that the NYT has a particularly easily classifiable range of *human interest*-related articles in that range.

Further analysis of the breakdown in Figure 6-11 also reveals that some categories obtain very high F-scores even with very little information. For instance, *sport* and *economy, business and finance* get F-scores in the high 70% and 60% respectively, with the latter even having a clear peak in F-score for length group 3 with otherwise little variation. Observing irregularities like these shows that there are more forces affecting the scores than the quantity of the information, and that the effects of adding more information are highly dependent on the category. Still, the macro-averaged scores attest the claim that longer articles generally are easier to classify, and that the most descriptive terms in a news article are probably located towards the beginning.

6.4 PERFORMANCE OVER TIME

One can argue that the uncertainty of a prediction from a model trained on historical data increases as time passes by. The following experiment investigates how this uncertainty affects the classifiers performance.

PREPARATION

The NYT Corpus contains articles spanning from 1987 to 2007. To get the most out of each time period, the dataset was first ordered chronologically by date, and then split into one training set, and ten equally sized validation and test sets, as shown in Table 6-2.

The split was done in such a way that each of the ten validation and test sets made up one third of the training set, and thus maintaining the properties of the standard split.

GROUP	DATE START	DATE END	ARTICLES
TRAIN	1987-01-01	1989-01-23	183986
TEST 1	1989-01-24	1990-07-21	61329
TEST 2	1990-07-22	1992-04-21	61329
TEST 3	1992-04-22	1994-05-22	61329
TEST 4	1994-05-23	1996-04-27	61329
TEST 5	1996-04-28	1998-05-04	61329
TEST 6	1998-05-05	2000-04-17	61329
TEST 7	2000-04-18	2002-01-18	61329
TEST 8	2002-01-19	2003-09-06	61329
TEST 9	2003-09-07	2005-06-02	61329
TEST 10	2005-06-03	2007-12-31	61329

TABLE 6-2 – TIME GROUPS

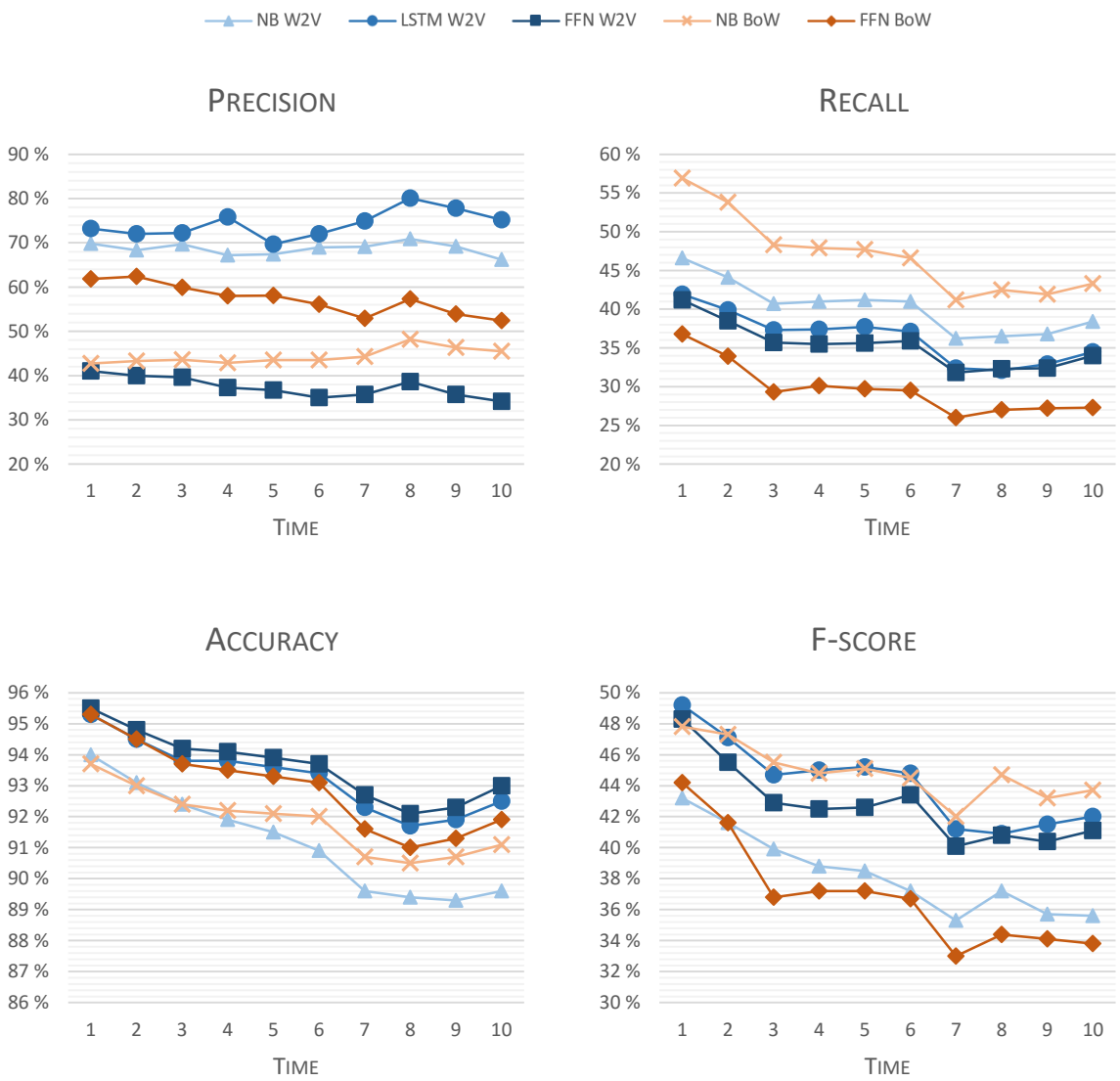


FIGURE 6-12 – MACRO-AVERAGED CLASSIFICATION PERFORMANCE OVER TIME

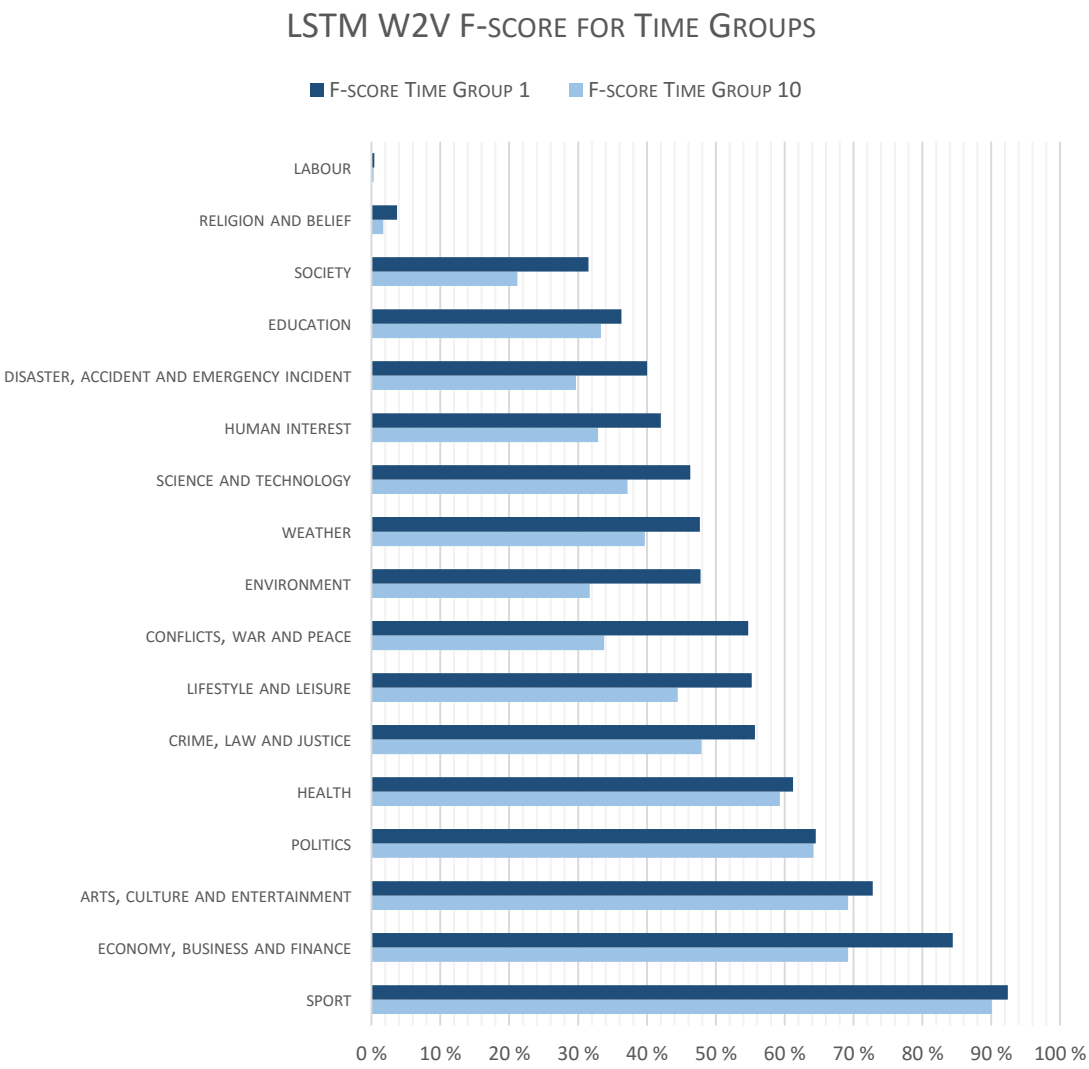


FIGURE 6-13 – CATEGORICAL BREAKDOWN OF F-SCORE BETWEEN TIME GROUP 1 AND TIME GROUP 10 FOR LSTM

RESULTS

Figure 6-12 shows how the macro-averaged scores develop over time. Common for all of the methods is that the overall F-score decays as time pass – mainly because of the decreasing recall. Accuracy naturally follows suit.

One possible reason for this behavior was thought to be that the available information the classifier is able to extract from the test sets in the distant future, is also decaying. However, when looking at the average number of annotations in response to the different time periods this is shown not to be the case. The development is shown quite clearly in Figure 6-14, where the available information in terms of annotations seems to actually *increase.* However, despite this increase in annotations, Figure 6-15 shows that the number of words remains rather constant. With other words, it is the *density* of annotations per article that increases. One plausible reason for the increased density is that Wikipedia, DBpedia's data source, may simply contain more information about recent entities than past.

Another possible reason for this decline may just be that some of the features assumed relevant for a given category in 1987-1989 – the span of the training set – simply shifted with time. This assumption is somewhat strengthened by looking at Figure 6-13, which shows the categorical breakdown of the F-score between the first and last time group for the LSTM classifier. Interestingly, *sport* – the by far most successful category, seemed also to decay the least over time, whereas *conflicts, war and peace* decayed the most.

This observation makes sense in the way that news articles about sporting events often follow the same style and lingo. On the other hand, articles revolving major, and perhaps world-spanning events like wars and global conflicts, may inadvertently pull in a significant number of entities previously unrelated to these topics. However, these assumption remains inconclusive as they were proven difficult to investigate.

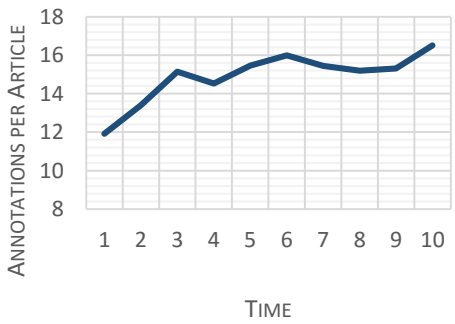


FIGURE 6-14 – AVERAGE NUMBER OF ANNOTATIONS PER ARTICLE FOR EACH TIME GROUP

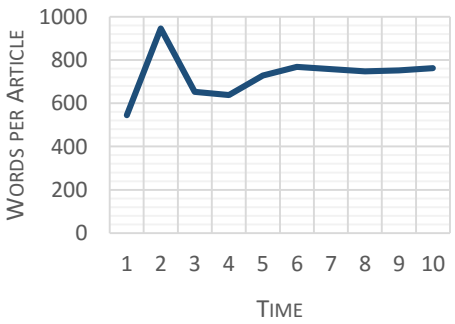


FIGURE 6-15 – AVERAGE NUMBER OF WORDS PER ARTICLE FOR EACH TIME GROUP

6.5 WORD EMBEDDINGS VERSUS BAG OF WORDS

Word embeddings are, as discussed in Section 2.2.3, a way of incorporating contextual information into the representation of words or entities. While the other experiments adhere to evaluating classification performance through a variety of scenarios and environments, the purpose of this experiment is to evaluate how an embedded approach compares to the traditional BoW approach.

PREPARATION

For this experiment, the full dataset was used. That is, 846 334 articles in the training set, and 282 111 in the test set. All models were then trained and tested using the standard split.

RESULTS

Figure 6-16 shows the overall macro-averaged score for all of the methods, with Word2Vec classifiers in blue, and BoW classifiers in red. First of all, what the macro-averaged scores shows is that there is no obvious winner. LSTM ranked the highest of the Word2Vec models with an F-score of 48 %, with NB BoW being only a single percent higher, at 49 %.

This subtle distinction is quite interesting as the Word2Vec vectors only contain 1000 features, far less than the BoW vectors which have a feature for every phrase – about 25 000 in total. With other words, the 1000 features in the Word2Vec vectors, are in this given experiment capable of conveying the same amount of information as the full-blown BoW vectors.

Looking at the example-based scores in Figure 6-17 there is at first glance little difference between the Word2Vec and BoW methods also here. Subset accuracy – the fraction of news articles which got *all of their* categories correctly assigned – is just below 40 % for both the best Word2Vec and BoW classifiers. *Hamming loss* – the fraction of incorrectly assigned labels – ranges in the mid 40's.

The biggest distinction is in the normalized label cardinality [5.4], but this is believed to be because of eagerness in the NB classifier (explained in Experiment 6.6), and not a property of the BoW representation. The same argumentation also holds for the Subset-One accuracy – the fraction of articles which had *at least one* category correctly assigned – which is 7 % higher for the NB BoW than LSTM W2V. Firing more shots increases the chances of hitting *something.*

Another interesting observation is that although both LSTM W2V and NB BoW got similar F-scores, the categorical breakdown in Figure 6-18 reveals that the similarity has exceptions. For instance, for the categories *labour, society* and *religion and belief* the difference in F-score is quite significant – as much as 20 % for *labour.*

So what is the deal with *labour*? Table 4-5 shows that half of the articles labeled *labour* are also labeled *economics* – the third best performing category overall. From a statistical standpoint, some of this should transfer into the labeling process for *labour*. This does however not seem to happen, and the actual reason for this behavior remain unresolved.

Either way, across the board, the embedded Word2Vec approach shows results on par with the traditional BoW approach from a macro perspective.

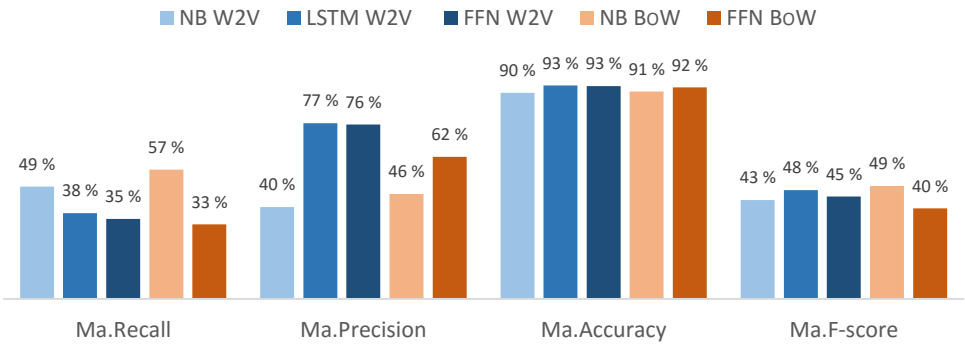


FIGURE 6-16 – MACRO-AVERAGED SCORES FOR THE FULL DATASET

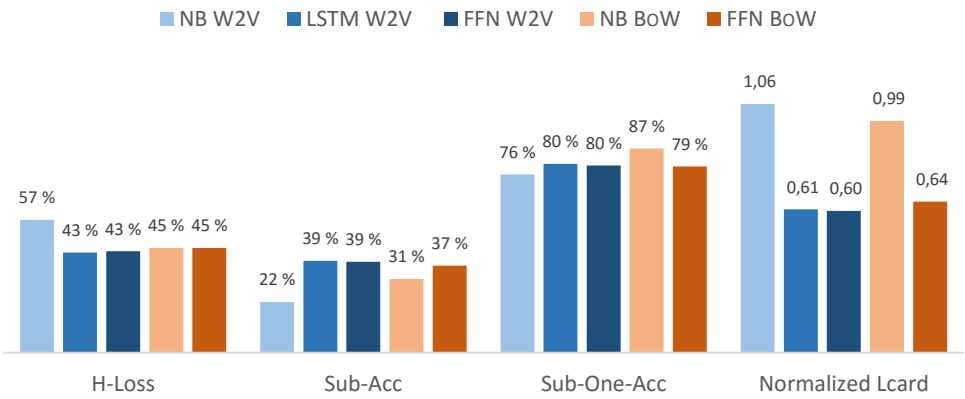


FIGURE 6-17 – EXAMPLE-BASED SCORES FOR THE FULL DATASET

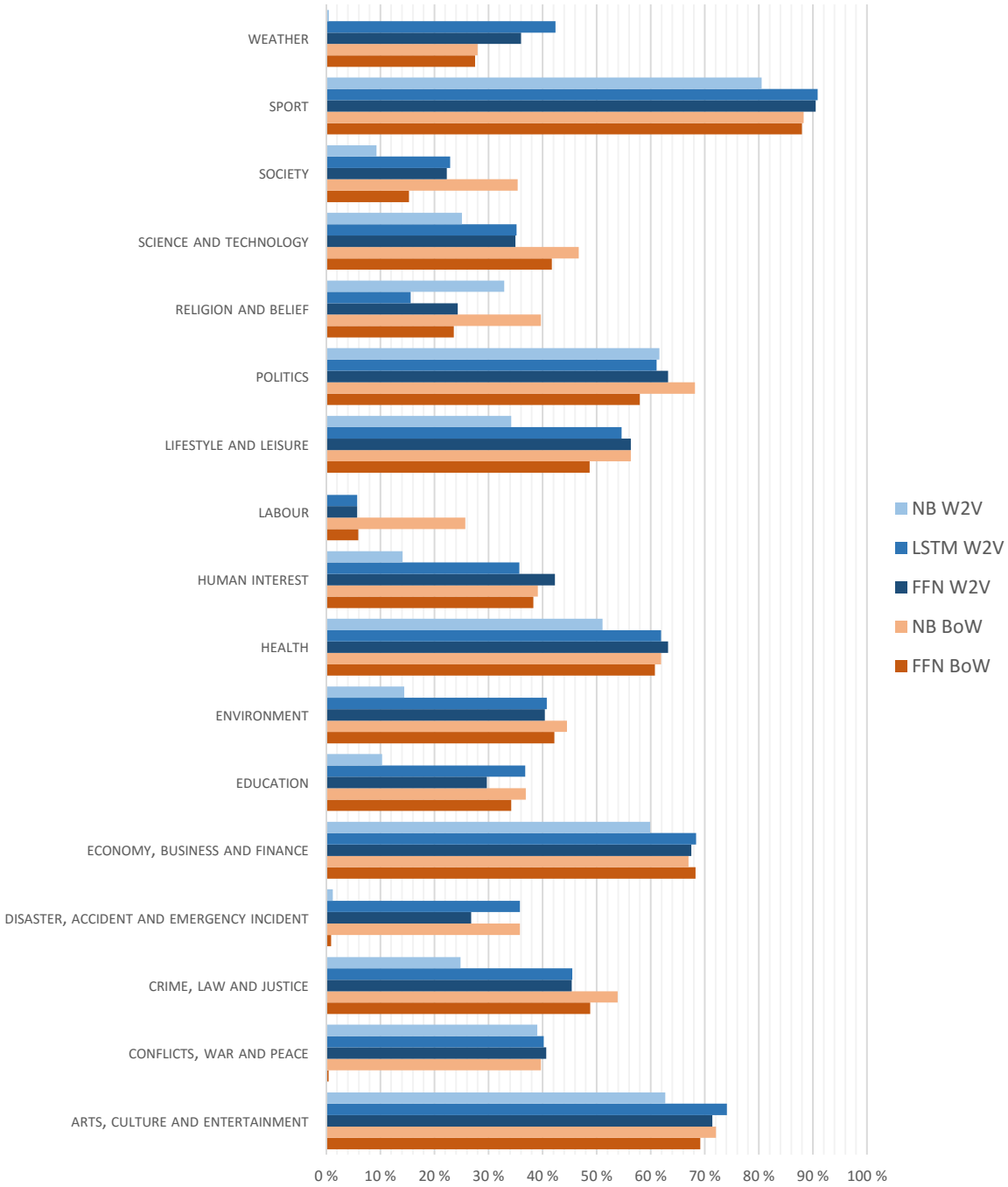


FIGURE 6-18 – F-SCORE BREAKDOWN BY IPTC CATEGORY

6.6 NAÏVE BAYES VERSUS DEEP LEARNING

While the preceding experiments have refrained from going into detail regarding the specific methods, the purpose of this last experiment is to do exactly that – and serve as a complement to the other results.

PREPARATION

This meta-experiment is based on the results from all of the preceding experiments.

RESULTS

Throughout all of the conducted experiments the perhaps most notable difference between the neural and the traditional Bayesian approach, is that NB generally sacrifices precision for recall. Another way to put this is that the NB more eagerly assigns labels – it has lower confidence. For the FFN and LSTM, the behavior seems to be the opposite. From the definition of the F-score in Section 2.7: If the F-score remains constant, a higher recall implies a higher number of false negatives. That is, news articles that *should have been* a given category, but that the classifier did not identify as such.

The takeaway from this is that the NB correctly labels news articles more often than the neural methods, at the expense of also misclassifying news articles at a higher rate. This relationship is possible to infer from the label cardinality in Figure 6-17, but is perhaps made even clearer with additional visual aid as shown in Figure 6-20. The diagram shows the total amount of annotations on the vertical axis, with every pillar representing the distribution of *TP, FN, FP* and *TN* for every category. The red areas represent mistakes made by the classifier, and the blue marks the successful predictions. By comparing the two charts it is possible to notice that the light red (false positives) sections are a lot bigger for NB than for LSTM. Conversely, the dark areas dominate the mistakes for the LSTM.

Another distinction between the neural and NB classifiers is the subset accuracy, which is consistently higher for the former. However, the reason for this can be inferred from the eagerness described above. Assigning more labels also results in a higher chance of error.

On a categorical level, the differences between NB and the neural approaches is more significant, as displayed in the categorical breakdown in Figure 6-18. Perhaps the most noticeable deviation is the results for *labour.* The neural methods are not able to reach an F-score beyond a mere 5% for this category, whereas NB BoW reaches 25 %. A similar, but opposite relationship is possible to spot for *weather,* where it is the NB BoW being inferior. Further investigation of the categorical breakdown uncovers numerous related observations. However, as mentioned in Experiment 6.5, the actual reason for this behavior remains unknown.

As for the neural approaches, one of the selling points for the LSTM is that it can work with ordered input. Still, throughout these experiments, the rather slim differences

between FFN W2V, NB W2V and LSTM W2V shows that, in this particular context, there is little to be gained from exploiting order. Other than that, the overall quality of the classification process seems to converge towards the same F-score.

In terms of computability, it is no secret that NB is a lot faster to train than the neural methods in terms of hours on the clock. However, how *fast* the methods learn in terms of training examples has been left mostly unexplored so far. An example of this development is shown in Figure 6-19, which plots the F-score for FFN W2V and NB BoW as a function of training examples. In addition, the plot also shows how the error in the FFN changes with training. As elaborated in Section 2.5.5, an ANN is trained by minimizing error, which in this case is inversely proportional to the F-score. This really just means that the network is *learning* something.

The perhaps most interesting observation to be made in Figure 6-19 is that while both NB BoW and FFN W2V learns quite a lot with very few examples, the FFN keeps on learning, while the NB BoW saturates almost immediately. While this example only shows the development for a single category, a similar relationship was found for other categories as well. It was also shown to be the case for the LSTM.

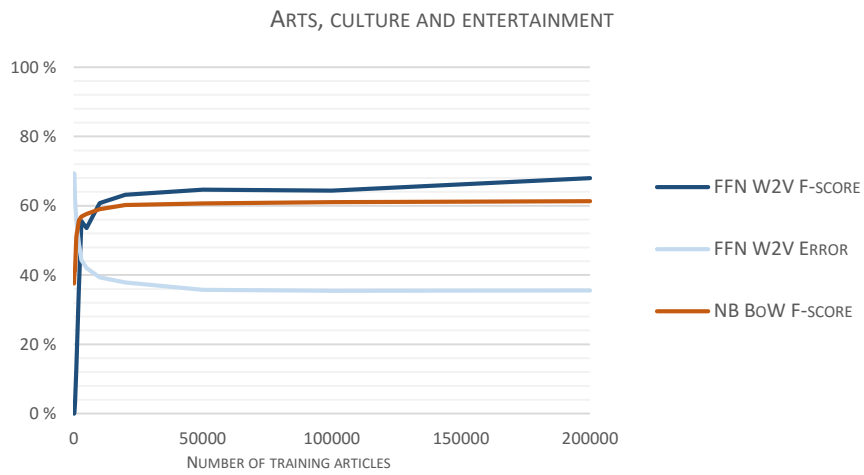


FIGURE 6-19 – LEARNING SPEED IN TERMS OF NUMBER OF EXAMPLES

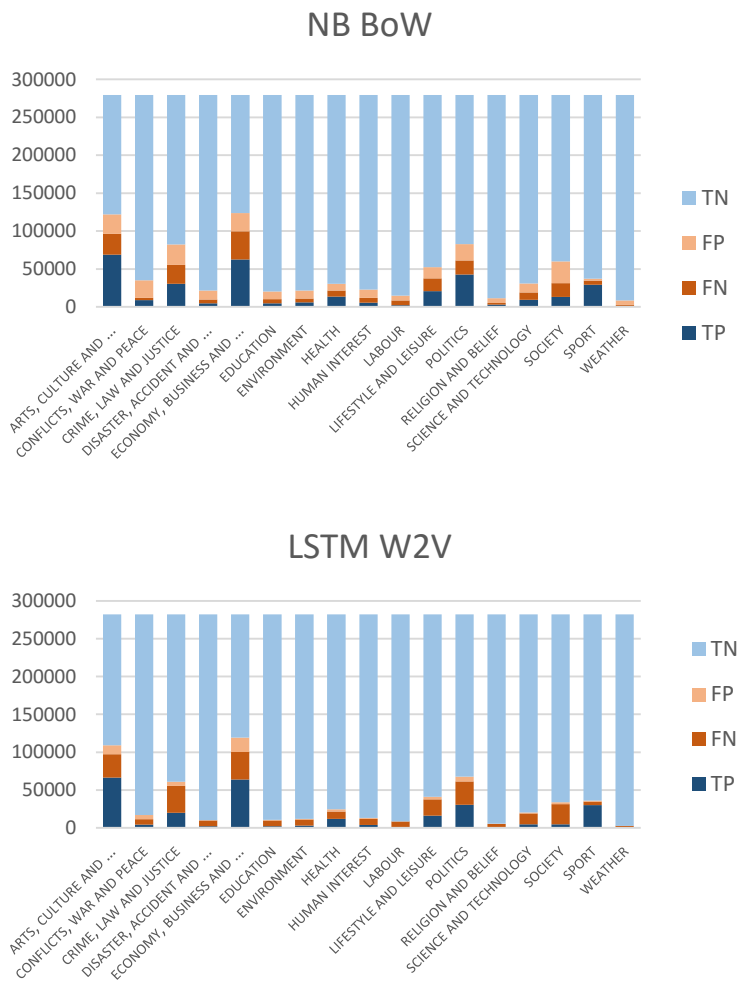


FIGURE 6-20 – CATEGORICAL DISTRIBUTION OF TP/FN/FP/TN FOR NB BOW AND LSTM W2V

6.7 WORDS VERSUS ANNOTATIONS

The purpose of this last experiment is to provide some closure to supplement the other experiments. The reason for its inclusion is just to give an intuition of how a "normal" approach would perform on the same dataset, where *normal* means that the classification is based on the entire body text and not solely on extracted annotations. Or to paraphrase: An approach that does not account for the multilingual principles which motivate the annotation-based methods.

As the approach used in this experiment is not among the main focal points of this thesis, only a quick presentation of the results follows.

PREPARATION

For this experiment, a 6th classifier was added: The *Traditional Naïve Bayes Bag of Words* (NB Trad). NB Trad is similar to the NB BoW, with the exception that it uses the actual *words* of from the news article as annotations, instead of just the extracted entities. Another modification done for this method was to apply *Porter Stemming* [2.3.1] before creating the annotations from the body text. Stemming, together with frequency filtering (done according to the procedures describe in Section 4.2.1), was done to reduce the dimensionality. Though, even with the reduction, the dimensionality of the word-feature space still ended up containing almost 100 000 features. The classifier is otherwise identical to NB BoW.

The dataset used for this experiment was the full corpus with the standard split.

RESULTS

As shown in the accumulated results in Figure 6-21, and in the categorical breakdown in Figure 6-22, is that the NB Trad has a very hard time learning the features of this dataset when using only *words* as a basis for its features. In fact, there are only *four* categories which the classifier manages to produce an F-score higher than zero.

The reason for these incredibly low F-scores is believed to originate from the same issues discussed in Experiment 6.2, which discussed supertype inclusion: The dimensionality of the feature vector combined with the imbalanced dataset, is simply too great to produce any statistically significant traction during the training phase.

What this experiment uncovers, is that the extracted entities not only provide a language-independent basis, but it also acts as a way of reducing the dimensionality, while at the same time improving the classifier.

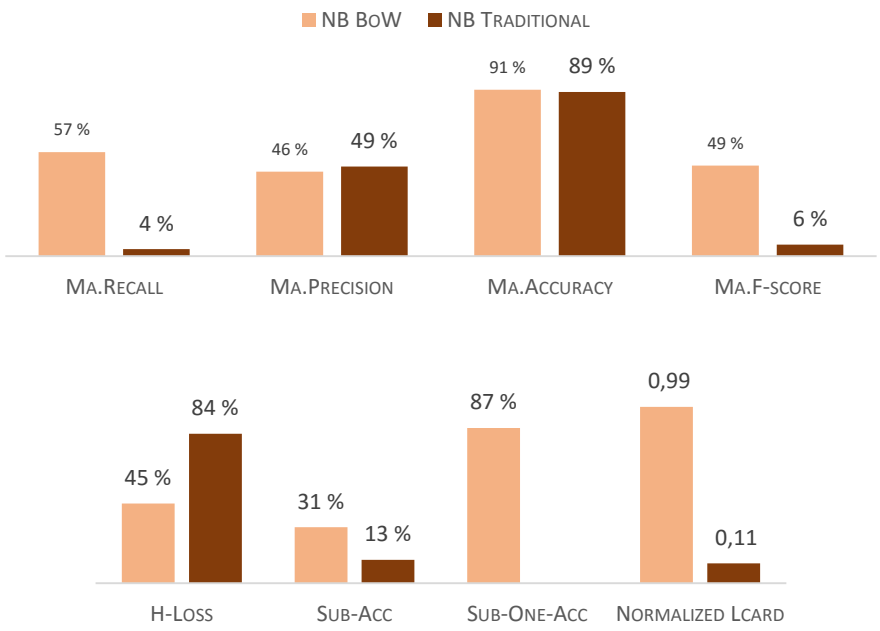


FIGURE 6-21 – BASELINE RESULTS FOR TRADITIONAL NAIVE BAYES

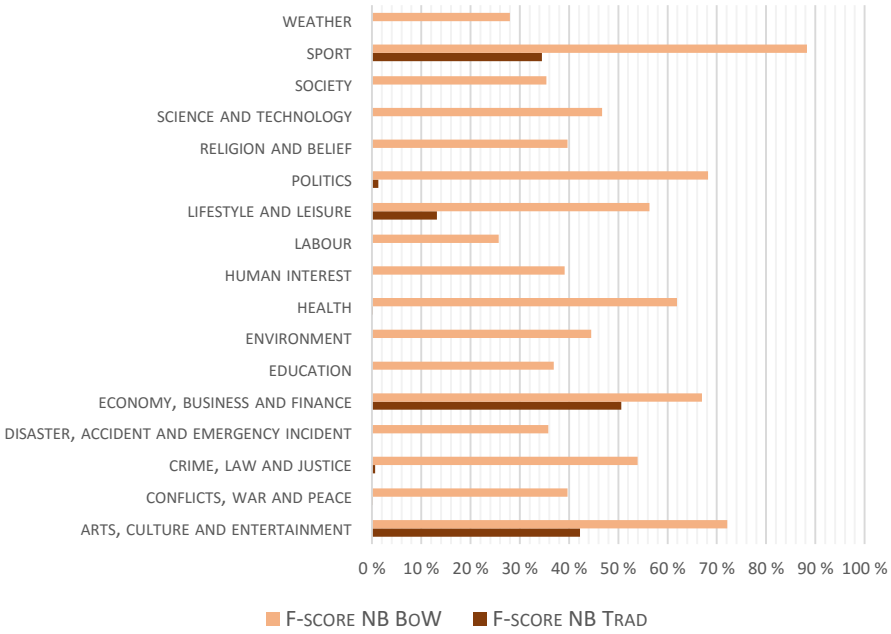


FIGURE 6-22 – CATEGORICAL BREAKDOWN OF TRADITIONAL NAIVE BAYES RESULTS

7 CONCLUSION

This final chapter of the report wraps up the thesis, with the primary motive being to shed light on the findings and the observations made. The first part of the chapter is dedicated to discussing the quality of the results, as well emphasizing the strengths and weaknesses of the system. This is then followed by a summarized assessment of the research questions. A brief summary of possible extensions for future work completes the chapter, and concludes the thesis.

7.1 DISCUSSION

This project has undertaken the ambitious task of designing a language agnostic and multi-label news article classification system, whose features are based solely on extracted DBpedia annotations. As the experiments from the preceding chapter uncover, the results are mixed – both in terms of the methodic classification quality, and performance across categories. While the focus of the experiments was to evaluate empiric results, the following section aims to spell out the strengths and weaknesses of the system from a practical standpoint.

RELATED RESEARCH

First of all, exploring related research for this thesis has been challenging. While there are a great number of relevant studies done in each of the individual fields explored in this project, relevant studies combining each approach the way it is done in this thesis are far and few in between. The most relatable family of research is probably sentiment and blog post classification, as mentioned throughout Chapter 3.

For this reason, it has been proven difficult to dig up an adequate comparison for the findings of this project, which in turn complicates the evaluation process. For instance, as mentioned in Section 3.4, most previous research in the news domain has used entirely different datasets, with wildly different characteristics. Also, the multi-label environment is often overlooked. Although this poses a challenge when evaluating the findings, it also shows that the chosen domain is in a preliminary state – which in turn promotes a bright future with great potential for the field as a whole.

It should also be noted that this project has attempted to cover *a lot of* ground. Getting a full overview of every little tidbit in related areas of research has been an inconceivable task. Still, I feel confident that the project was successful in covering the most relevant elements, while simultaneously providing leeway for future research.

THE DATASET AND DOMAIN

Even though the NYT corpus is sheer in size, the quality of the news articles themselves is varied. As the corpus contains more or less anything published in NYT over the span of 20 years, it includes anything from letters to the editor, death notices, corrections,

promotional content and even a significant number of *lists* (e.g. all books published in years 2000). Not only does this "noise" complicate the training process, but it also raises questions regarding the quality of the classification, as the empiric results reflect how well the classifier manages to classify the collective *contents* of a news aggregator, and not actual news articles.

Similar issues are also spotted in the tagging of the articles. Although most articles, as shown in Figure 4-14, were successfully attributed with IPTC Media Topic categories, the original descriptors were subject to manual tagging schemes. In short, this means that there is not necessarily any strict consistency in what constitutes a *labour* or *weather* article – and even less so when using only extracted annotations (i.e. nouns) as identifying features. For example, there are several cases where different articles end up with an identical set of extracted annotations, yet attributed with completely different categories, ultimately making it impossible for the classifier to get them all right.

Another property of the corpus, and perhaps the media industry in itself, is that the overall category distribution is absurdly skewed [Figure 4-15]. There are simply more news articles related to *politics*, *economics lifestyle* and *arts*, than there are news articles covering *weather* and *religion* – making attaining the necessary amount of training data for the smaller categories difficult. The end result is, as several observations in this project substantiate, that the quality of the classification process inevitably suffers from this lack of information.

Nonetheless, it is believed that simply increasing the quantity of the dataset would probably not have sufficed in the long run in terms of language independence. This is primarily because everything in the NYT corpus is *American*. As shown in Table 4-4, this is something that profoundly influences both the diversity of the annotations, but also their corresponding relevance with respect to the different categories.

The same conclusion can in a sense be drawn for the online ontologies like DBpedia and Wikidata, which at the time of writing predominantly covers concepts and entities rooted in Western values – especially in terms of celebrities and pop culture. On the other hand, these ontologies are a live and growing resource, with open and active communities – while the corpus remains static.

CLASSIFICATION

Throughout Chapter 6, the performance of the deep learning inspired methods has been more or less indistinguishable from the in comparison trivial NB classifier. Does this imply that deep learning does not work and should be immediately scrapped in favor of the much more efficient NB? Probably not. Truth be told: As emphasized in Chapter 5, the difficulty of configuring and training ANNs most likely has constrained the real potential of deep learning. However, what the results also show, is that achieving satisfactory results with complicated neural nets, is within reach even for someone otherwise fresh in the field. Furthermore, the research also shows that using n-binary classifiers is a viable choice even when using neural methods, which further strengthens the claims in related research, discussed in Section 3.1.

As far as the actual classification quality goes, the research conducted in this thesis demonstrate a far-reaching point: That it is possible to classify news articles only through consideration of extracted language-independent noun-based entities, with some categories like *sports* reaching an F-score as high as 91% [Figure 6-18]. On a related note, it remains unclear whether using proper nouns as the only means of information for news articles, is a sustainable approach across the board. Certain categories, like *economics, business and finance* did not outperform *sport*, despite being more than three times the size, regarding training data.

In terms of categories, IPTC Media Topic has been proven a valuable resource when classifying news articles in a language-independent manner. The extensive breadth covered by the five ontologically cataloged layers, enables a very flexible and convenient guideline for classifying a wide variety of news articles.

TECHNICAL CHALLENGES

Aside from being an ambitious project from a research perspective, the technical difficulties faced throughout this project have been at least as much of a challenge. Ranging from troubleshooting distributed environments, to customizing the garbage collector to avoid race conditions when concurrently training neural networks, to having a job crash after six days because of bugs in a third party framework – this project has at times felt like a depth-first search through an ocean of issues. And that is only the technical side. Then there is the deep learning side.

Admittedly, this project started off with the naïve assumption that deep learning is some sort of silver bullet in machine learning – a tool that can just be applied to any given problem, and it will just work. The harsh reality is quite different. Deep learning is *hard*. It is in fact *very hard*. And this is something that has become more than crystal clear throughout this project.

Section 5.5 scratched the surface of hyperparameter tuning, and mentioned that *Random Search* was more or less how the neural networks were configured and tuned for this project. While this still holds, what it means in practice is having eight servers running 24/7 for weeks trying to find something that *might* work. Not only is it a challenge to manage all of these servers to begin with, but keeping track of parameters that may work, what to explore and so on, is in itself a struggle. Juggling all of the results, configurations and different models have similarly posed a major challenge, and many experiments had to be run several times simply because result got *lost*.

Finally, training neural nets with the chosen frameworks in a distributed environment like Apache Spark was also proven a lot more difficult than expected. Partially because of the immaturity of the deeplearning4j framework, and a lack of knowledge in distributed computing. Spark is not always as straightforward as one might seem.

Although the technical challenges faced over the course of this project were numerous, and to a certain extent crippled the deep learning aspects of the thesis, the lessons learned are invaluable.

7.2 CONCLUDING REMARKS

After having gained considerable insight about the field, both theoretically, and experimentally, it is time to address the research questions and emphasize the key findings of this thesis. The answers in this section are meant to serve as a concise and to-the-point summary, complementing the extensiveness of the preceding chapters.

RQ1 ***How does deep learning compare to traditional machine learning techniques in news categorization in terms of classification quality?***

While there are no doubts about the potential of deep learning, this study's neural techniques did not manage to significantly surpass simple traditional methods like Naïve Bayes in terms of classification quality. In fact, the classification quality of the best performing methods from either domain was more or less indistinguishable on the macro level.

However, the reason for these results is believed to be related to a combination of sub-optimal hyperparameter configurations and insufficient training on the part of the deep learning methods, rather than them being inferior methods for this task.

Looking at the deep learning approaches in isolation, the LSTM – which is capable of modeling ordered input – performed marginally better than the FFN in terms of F-score (+3%). This finding is in sync with previous research emphasizing the importance of accounting for word order in text classification [104].

Although some of the findings in this study are somewhat conflicting with the recent hype in deep learning, discussed in Section 2.5 and 3.3, the attempted approaches were not exhaustive. For this reason, conclusions based on these findings remain undecided, and are subject to further research.

RQ2 ***How does word embeddings compare to a bag-of-word approach in news categorization?***

For this thesis, using word embedded feature construction through utilization of a pre-trained Word2Vec, was shown to be comparable to bag-of-words regarding classification quality. Subsequently, for all of the deep learning-based classifiers used, Word2Vec gave superior results over bag-of-words in all cases. This supports the claim made by [17], discussed in Section 3.3, that pre-trained Word2Vec models are good universal feature extractors.

In addition, the study provides evidence showing that creating normalized document vectors by first performing scalar multiplication with TF-IDF weights, followed by adding the vectors together, yields little to no loss in classification quality compared to representing features as a chain of ordered word vectors.

Furthermore, the models using the accumulated document vectors were also, in general, the fastest models to train – both because the document vectors could be

pre-computed and because the dimensionality of the document vectors generally is significantly lower than the corresponding bag-of-word vectors.

RQ3 ***How does the number of annotations and article lengths affect the accuracy of news categorization?***

The conducted experiments showed that the macro-averaged F-score increased proportionally with the article length up to around 600 words (≈ 14 annotations), at which point the F-score stabilized.

On a categorical level, this limit is less clear. Some categories, like sport and economics & finance, performed very well even at lengths below 136 words (≈ 3 annotations).

Although 600 words is not a golden number for all categories, there is a consensus that there is an upper limit at which point the F-score does no longer increase.

RQ4 ***How does moving through time affect news categorization performance, and what is the relevance of up-to-date training data?***

For this thesis, moving through time was shown to slightly degrade the classification quality, even though the number of annotations increased with time. The article length did not change significantly, thus promoting the idea that extracting annotations from older news articles is less trivial than recent ones.

As the F-score decreased even with the increase of annotations, the impact of moving through time is assumed to be greater than what is observed in the conducted experiments.

Like with the article length experiment, the categorical breakdown of the results shows that categories are impacted with varying degrees of significance when moving through time. Some categories, like sport (-2%) and politics (-1%), remain nearly unaffected even on test data ten years in the future, whereas conflicts, war and peace (-21%) and environment (-16%) are affected the most.

RQ5 ***How does incorporating ontologically related supertypes affect the quality of news categorization?***

The conducted experiments show no evidence to promote the hypothesis that incorporating ontologically related supertypes into the classification process has any positive effects on classification performance. In fact, incorporating supertypes in several cases *reduced* performance, both in terms of classification quality and computational demands, as it expanded the feature space without providing any *new* information.

Thus, the observations made further support the claims discussed in Section 3.2 and [3], stating that adding ontologically related *synsets* has little to no effect when classifying textual content.

RQ6 ***How does noun-based feature extraction of ontological data promote language agnostic news classification?***

Language independence has been the underlying theme throughout the entire project, and the experiments and the proposed preprocessing pipeline points in favor for a noun-based approach based on extracted ontological annotations.

The study shows that it is possible, only by using noun-based language-independent DBpedia/Wikidata entities and top-level IPTC Media Topics categories, to classify certain categories covering topics like sports, economics, and arts & entertainment with high accuracy. These findings further support the conclusions made in [96] and discussed in Section 3.2, that using ontologically extracted entities forms a robust basis for feature construction.

Although the quality of the classification is quite inconsistent between categories, the study provides evidence that this is a property of the dataset itself, and not the proposed methods. It is therefore assumed that the inconsistencies could be evened out through a more carefully tuned feature construction process. For instance, by better promoting the statistical significance of key entities, or by reducing the imbalance between categories.

RQ7 ***What are the important limitations or challenges of deep learning in news categorization, and under what circumstances are the additional complexities of deep learning justified?***

Regarding computational requirements, the deep learning methods, especially LSTM, generally require a lot more resources compared to traditional methods like Naïve Bayes. This is both in terms of CPU cycles and memory, but also in training time. For this thesis, the increase in training time was sometimes several orders of magnitude – from the 30 minutes with Naïve Bayes, to several days with an LSTM. Subsequently, prediction time was found to increase proportionally with the complexity of the network topology. In addition, deep learning methods are proven extremely hard to train beyond the limits of a linear classifier. Furthermore, tuning hyperparameters and figuring out a suitable topology is an equally time-consuming, and error-prone process.

While not dismissing the potential in deep learning for the news article domain, general takeaway from the experiences gained throughout this study is: (i) One should not underestimate the power of traditional methods and that (ii) deep learning should only be a consideration if the simpler approaches do not meet the requirements, and if project has sufficient time and resources to survive probable failures, with no immediate guarantee for success.

7.3 FURTHER WORK

The core motivational driver behind this project was to facilitate the automation of reliable and language-independent news classification. While the findings made throughout this project are significant, the field of language agnostic multi-label news classification remains largely unexplored. As a consequence, the number of possible extensions for this research is plentiful, to say the least. This final section of the report is therefore dedicated to a brief summary of the most natural courses of continuation.

IMPROVE NEURAL TOPOLOGY AND HYPERPARAMETERS

Perhaps the most obvious extension of this project is simply to try out and examine more configurations and topologies. Although this project did not succeed in significantly surpassing traditional methods, it is still believed that the hyperparameters and neural topologies could be improved to achieve higher classification quality.

In addition, it would also be fascinating to see how a convolutional network would perform on the same classification task tackled throughout these experiments.

MULTILINGUALISM IN PRACTICE AND STANDARDIZED DATASETS

Although the theme for this thesis has been multilingualism, the reference language actually used has been limited to English. However, the proposed system should in theory work for other languages, given that the DBpedia/Wikidata annotations can be extracted, and that labeled datasets exist.

Exploring other languages in practice, would both be providing a new perspective on the news article classification problem, and significantly broaden the areas of application for the framework itself. This could also be extended to explore cultural differences in both news articles and text in general.

Furthermore, it would also be very useful for the research context to redo the experiments with a more frequently used dataset, like the Reuter datasets mentioned in Section 3.4.2. Not only would this offer a fresh viewpoint, but it would also make it a lot easier to evaluate the results in general.

COMBINE, MULTIPLY AND CONQUER

The experiments show that one of the strongest selling points for the well-performing categories is that they have a huge number of distinct annotations, while still maintaining a high term-frequency throughout the corpus [Figure 6-7]. Or put another way, too many unique annotations, leads to a reduction in statistical significance. One possible way to accumulate and utilize this otherwise lost information, could be to combine or cluster these terms – possibly by joining them with their corresponding ontological supertypes.

Another property of the well-performing categories is that they normally inherit less imbalance between true positives and true negatives. It is reasonable to believe that the

inferior performance of certain categories could be improved by balancing these differences, possibly by sub- or super-sampling of the dataset.

Integrating and experiment with the above strategies are natural succeeding steps in the search for higher classification quality.

INCORPORATING CORRELATIONAL DEPENDENCIES

Every approach carried out in this project has ignored correlations between categories. However, as shown in Table 4-5 it is clear that there is quite a bit of leverage to be utilized by exploiting correlational dependencies. For instance, the table indicates that the probability of an article being *economy* given that it got tagged *disaster*, is as much as 56 %. Creating a classification system that could account for this information certainly has the potential to improve classification quality.

REFERENCES

[1] P. J. Hayes, L. E. Knecht and M. J. Cellio, "A news story categorization system," *ANLC '88 Proceedings of the second conference on Applied natural language processing,* pp. 9-17 , 1988.

[2] B. Masand, G. Linoff and D. Waltz, "Classifying news stories using memory based reasoning," *SIGIR '92 Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval,* pp. 59-65, 1992.

[3] A. Hotho, S. Staab and G. Stumme, "Ontologies improve text document clustering," *Data Mining, 2003. ICDM 2003,* pp. 541 - 544, Nov 2003.

[4] Q. Luo, E. Chen and H. Xiong, "A semantic term weighting scheme for text categorization," 21 April 2011.

[5] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Computing Surveys,* vol. 34, no. 1, pp. 1-47, 2002.

[6] C. Apté, F. Damerau and S. M. Weiss, "Towards Language Independent Automated Learning of Text Categorization," *Proceeding SIGIR '94 Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval,* pp. 23-30, 1993.

[7] J. Harris, "Messing around with metadata," New York Times, 27 10 2007. [Online]. Available: http://open.blogs.nytimes.com/2007/10/23/messing-around-with-metadata/. [Accessed 03 08 2015].

[8] J. M. G. Hidalgo, M. d. B. Rodríguez and J. C. C. Pérez, "The Role of Word Sense Disambiguation in Automated Text Categorization," *Lecture Notes in Computer Science,* vol. 3513 , pp. 298-309, 2005.

[9] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition,* vol. 5(2), pp. 199-220, 1993.

[10] Z. Elberrichi, A. Rahmoun and M. A. Bentaalah, "Using WordNet for Text Categorization," *The International Arab Journal of Information Technology, Vol. 5, No. 1,* January 2008.

[11] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inform Process. Man.,* vol. 24, no. 5, pp. 513-523, 1988.

[12] G. de Melo and S. Siersdorfer, "Multilingual Text Classification Using Ontologies," *Advances in Information Retrieval,* vol. 4425, pp. 541-548, 2007.

[13] H. Bacan, I. S. Pandzic and D. Gulija, "Automated News Item Categorization," *Proceedings of the 19th Annual Conference of The Japanese Society for Artificial Intelligence, Springer-Verlag, Kitakyushu,* pp. 251-256, 2005.

[14] E. Mozzherina, "An Approach to Improving the Classification of the New York Times Annotated Corpus," *Knowledge Engineering and the Semantic Web,* vol. 394, pp. 83-91, 2013.

[15] M. Baroni, G. Dinu and G. and Kruszewski, "Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors," *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics,* vol. 1, pp. 238-247, 2014.

[16] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," *CoRR,* vol. abs/1310.4546, 2013.

[17] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *CoRR,* vol. abs/1408.5882, 2014.

[18] T. Mikolov, W.-t. Yih and G. Zweig, "Linguistic Regularities in Continuous Space Word Representations," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*, Association for Computational Linguistics, 2013.

[19] S. Vijayarani, M. J. Ilamathi and M. Nithya, "Preprocessing Techniques for Text Mining," *International Journal of Computer Science & Communication Networks,* vol. 5, no. 1, pp. 7-16.

[20] G. Forman, I. Guyon and A. Elisseeff, "An extensive empirical study of feature selection metrics for text classification," *Journal of Machine Learning Research,* vol. 3, pp. 1289-1305, 2003.

[21] M. F. Porter, "An algorithm for suffix stripping," *Program* , vol. 14, no. 3, pp. 130-137, 1980.

[22] L. D. Baker and A. K. McCallum, "Distributional clustering of words for text classification," *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development,* pp. 96-103, 1998.

[23] C. Silva and B. Ribeiro, "The importance of stop word removal on recall values in text categorization," *Neural Networks,* vol. 3, pp. 1661-1666, 2003.

[24] ranks.nl, "Stop Word Lists," ranks.nl, [Online]. Available: http://www.ranks.nl/stopwords. [Accessed 12 10 2015].

[25] "List of English Stop Words," xpo6, 03 10 2009. [Online]. Available: http://xpo6.com/list-of-english-stop-words/. [Accessed 12 10 2015].

[26] H. Saif, M. Fernandez, Y. He and H. Alani, "On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter.," *LREC 2014, Ninth International Conference on Language Resources and Evaluation,* p. 810–817, 2014.

[27] K. Toutanova and C. D. Manning., "Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger," *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000),* pp. 63-70, 2000.

[28] C. Manning and H. Schuetze, Foundations of Statistical Natural Language Processing, The MIT Press, 1999.

[29] X. Li and Y. Guo, "Active Learning with Multi-Label SVM Classification," *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence,* 2013.

[30] M.-L. Zhang and Z.-H. Zhou, "A Review on Multi-Label Learning Algorithms," *IEEE, Transactions on KNowledeg and Data Engineering,* vol. 99, 2013.

[31] A. Halevy, P. Norvig and F. Pereira, "The Unreasonable Effectiveness of Data," *IEEE Intelligent Systems, vol.24,* pp. 8-12, March/April 2009.

[32] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," *Lecture Notes in Computer Science,* vol. 1398, pp. 4-15, 2005.

[33] J. Hovold, "Naive Bayes Spam Filtering Using Word-Position-Based Attributes," *In Conference on Email and Anti-Spam,* 2004.

[34] Z. Wei, H. Zhang, Z. Zhang, W. Li and D. Miao, "A Naive Bayes Multi-Label Classification Algorithm," *International Journal of Advanced Intelligence,* vol. 3, no. 2, pp. 173-188, 2011.

[35] V. Narayanan, I. Arora and A. Bhatia, "Fast and accurate sentiment classification using an enhanced Naive Bayes model," *Intelligent Data Engineering and Automated Learning IDEAL 2013,* vol. 8206, pp. 194-201, 2013.

[36] I. Rish, "An empirical study of the naive bayes classifier," 2001.

[37] J. D. M. Rennie, L. Shih and J. T. a. D. R. Karger, "Tackling the Poor Assumptions of Naive Bayes Text Classifiers," *In Proceedings of the Twentieth International Conference on Machine Learning,* pp. 616-623, 2003.

[38] T. Joachims, "Text categorization with Support Vector Machines: Learning with many relevant features," *Machine Learning: ECML-98,* vol. 1398, pp. 137-142, 2005.

[39] D. G. Underhill, "Exploring Dimensionality Reduction for Text Mining," *Trident Scholar project report,* no. 362 , 2007.

[40] O. Sutton, "Introduction to k Nearest Neighbour Classification and Condensed Nearest Neighbour Data Reduction," February 2012.

[41] D. Wettschereck, D. W. Aha and T. Mohri, "A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms," *Artificial Intelligence Review,* vol. 11, no. 1-5, pp. 273-314, 1997.

[42] Y. Zhou, Y. Li and S. Xia, "An Improved KNN Text Classification Algorithm Based on Clustering," *Journal of Computers,* vol. 4, no. 3, pp. 230-237, 2003.

[43] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognition,* vol. 40, no. 7, p. 2038–2048, 2007.

[44] N. Yadav, A. Yadav and M. Kumar, An Introduction to Neural Network Methods for Differential Equations, Springer, 2015.

[45] F. Rosenblatt, Principles of neurodynamics: perceptrons and the theory of brain mechanisms, Spartan Books, 1962.

[46] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature,* vol. 323, no. 6088, pp. 533-536, 1986.

[47] K. Hornik, M. B. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks,* vol. 2, no. 5, pp. 359-366, 1989.

[48] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, pp. 436-444, 2015.

[49] "Analysis of different activation functions using back propagation neural networks," *Journal of Theoretical and Applied Information Technology,* vol. 47, no. 3, 2013.

[50] M. A. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.

[51] Y. LeCun, I. Kanter and S.A.Solla, "Second-order properties of error surfaces: learning time and generalization," *Advances in Neural Information Processing Systems,* vol. 3, pp. 918-924, 1991.

[52] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics,* 2010.

[53] I. Goodfellow, Y. Bengio and A. Courville, Deep learning, MIT Press, 2016.

[54] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., 2012, pp. 1097-1105.

[55] C. Bishop, Neural Networks for Pattern Recognition, Oxford: Clarendon Press, 1995.

[56] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean and A. Ng, "Building high-level features using large scale unsupervised learning," *International Conference in Machine Learning,* 2012.

[57] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE,* vol. 87, no. 9, pp. 1423-1447, 1999.

[58] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE,* vol. 78, no. 10, pp. 1550-1560, 1990.

[59] Y. LeCun, L. Bottou, G. B. Orr and K. -R. Müller, "Efficient BackProp," *Lecture Notes in Computer Science,* vol. 1524, pp. 9-50, 2002.

[60] M. Zinkevich, M. Weimer, L. Li and A. J. Smola, "Parallelized Stochastic Gradient Descent," *Advances in Neural Information Processing Systems 23,* pp. 2595-2603, 2010.

[61] Y. LeCun, "A Theoretical Framework for Back-Propagation," *Proceedings of the 1988 Connectionist Models Summer School,* pp. 21-28, 1988.

[62] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[63] H. T. Siegelmann, "Computation Beyond the Turing Limit," *Science,* vol. 268, no. 5210, pp. 545-548, 1995.

[64] R. Pascanu, T. Mikolov and Y. Bengio, "On the difficulty of training Recurrent Neural Networks," *CoRR,* vol. abs/1211.5063, 2012.

[65] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[66] I. Sutskever, J. Martens, G. E. Dahl and G. E. Hinton, "On the importance of initialization and momentum in deep learning," *Proceedings of the 30th International Conference on Machine Learning (ICML-13),* vol. 28, no. 3, pp. 1139-1147, 2013.

[67] K. He, X. Zhang, S. Ren and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on," *CoRR,* vol. abs/1502.01852, 2015.

[68] H. Robinds and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics,* p. 400–407, 1951.

[69] C. Darken, J. Chang and J. Moody, "Learning rate schedules for faster stochastic gradient search," *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop,* p. 1–11, 1992.

[70] J. Duchi, E. Hazan and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research,* vol. 12, p. 2121–2159, 2011.

[71] G. Hinton, "Neural Networks for Machine Learning, Lecture 6e.," [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. [Accessed 30 03 2016].

[72] N. Srivastava, G. r. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research ,* vol. 15, pp. 1929-1958, 2014.

[73] W. Zaremba, I. Sutskever and O. Vinyals, "Recurrent Neural Network Regularization," *CoRR,* vol. abs/1409.2329, 2014.

[74] M. Dash and H. Liu, "Feature selection for classification," *Intelligent data analysis,* vol. 1, no. 3, pp. 131-156, 1997.

[75] A. Ilin and T. Raiko, "Practical approaches to principal component analysis in the presence of missing values," *Journal of Machine Learning Research,* vol. 11, pp. 1957-2000, 2010.

[76] C. Olah, "Visualizing Representations: Deep Learning and Human Beings," 16 01 2015. [Online]. Available: http://colah.github.io/posts/2015-01-Visualizing-Representations/. [Accessed 02 04 2016].

[77] L. v. d. Maaten and G. Hinton, "Visualizing High-Dimensional Data Using t-SNE," *Journal of Machine Learning Research ,* vol. 9, pp. 2579-2605, 2008.

[78] Y. LeCun, C. Cortes and C. J. Burges, "THE MNIST DATABASE of handwritten digits," [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed 24 06 2016].

[79] A. Sun and E.-P. Lim, "Hierarchical Text Classification and Evaluation," *Data Mining, 2001. ICDM 2001, Proceedings 2001 IEEE International Conference on Data Mining,* pp. 521 - 528, 2001.

[80] D. D. Lewis, "Evaluating Text Categorization," *In Proceedings of Speech and Natural Language Workshop,* pp. 312-318, 1991.

[81] R. E. Sharpire and V. Singer, "A Boosting-based System for Text Categorization," *Machine Learning,* vol. 39, pp. 135-168, 2000.

[82] D. Tikk and G. Biró, "Experiments with multi-label text classifier on the Reuters collection," *Proceedings of the international conference on computational cybernetics (ICCC 03),* 2003.

[83] R. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Proceedings of the 11th Annual Conference on Computational Learning Theory, New York,* pp. 80-91, 1998.

[84] X. Li, S. Szpakowicz and S. Matwin, "A WordNet-based Algorithm for Word Sense Disambiguation," *In Proceedings of the 14th International Joint Conference on Artificial Intelligence,* pp. 1368-1374, 1995.

[85] N. Bel, C. H. A. Koster and M. Villegas, "Cross-Lingual Text Categorization," 2003.

[86] J.-J. R. A. C. a. D. L. d. I. García-Adeva, "Multilingual approaches to text categorisation," *CEPIS promotes,* vol. 43, 2005.

[87] J. S. Olsson, D. W. Oard and J. Hajič, "Cross-language text classification," *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '05),* pp. 645-646, 2005.

[88] L. Rigutini, M. Maggini and B. Liu, "An EM based training algorithm for cross-language text categorization," *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05),* pp. 529-535, 2005.

[89] T. Mansuy and R. J. Hilderman, "A characterization of wordnet features in Boolean models for text classification," *AusDM '06 Proceedings of the fifth Australasian conference on Data mining and analytics,* vol. 61, pp. 103-109, 2006.

[90] D. Yogatama, D. Gillick and N. Lazic, "Embedding methods for fine grained entity type classification," *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL,* pp. 26-31, 2015.

[91] K. Nebhi, "Named Entity Disambiguation using Freebase and Syntactic Parsing," in *SEMWEB*, 2013.

[92] Z. Zheng, X. Si, F. Li, E. Y. Chang and X. Zhu, "Entity Disambiguation with Freebase," *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology,* vol. 1, pp. 82-89, 2012.

[93] S. D. Husby and D. Barbosa, "Topic Classification of Blog Posts Using Distant Supervision," *Proceedings of the Workshop on Semantic Analysis in Social Media,* pp. 28-36, 2012.

[94] A. Graves and N. Jaitly, "Towards End-To-End Speech Recognition with Recurrent Neural Networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, JMLR Workshop and Conference Proceedings, 2014, pp. 1764-1772.

[95] L. Johnston, "Using LSTM Recurrent Neural Networks for Music Generation," 2016.

[96] O. Vinyals, A. Toshev, S. Bengio and D. Erhan, "Show and Tell: A Neural Image Caption Generator," *CoRR,* vol. abs/1411.4555, 2014.

[97] S. Venugopalan, M. Rohrbach, J. Donahue, R. J. Mooney, T. Darrell and K. Saenko, "Sequence to Sequence - Video to Text," *CoRR,* vol. abs/1505.00487, 2015.

[98] I. Sutskever, O. Vinyals and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *CoRR,* vol. abs/1409.3215, 2014.

[99] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl and V. Svetnik, "Deep Neural Nets as a Method for Quantitative Structure–Activity Relationships," *Journal of Chemical Information and Modeling ,* vol. 55, no. 2, pp. 263-274, 2015.

[100] M. K. K. Leung, H. Y. Xiong, L. J. Lee and B. J. Frey, "Deep learning of the tissue-regulated splicing code," *Bioinformatics,* vol. 30, no. 12, pp. 121-129, 2014.

[101] A. Graves, "Generating Sequences With Recurrent Neural Networks," *CoRR,* vol. abs/1308.0850, 2013.

[102] A. Timmaraju and V. Khanna, "Sentiment Analysis on Movie Reviews Using Recursive and Recurrent Neural Network Architectures," 2015.

[103] A. Graves, Supervised Sequence Labelling with Recurrent Neural Networks, Springer, 2012.

[104] R. Johnson and T. Zhang, "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks," *CoRR,* vol. abs/1412.1058, 2014.

[105] D. D. Lewis, Y. Yang, T. Rose and F. Li, "RCV1: A New Benchmark Collection for Text Categorization Research," *Journal of Machine Learning Research,* vol. 5, pp. 361-397, 2004.

[106] M.-R. Amini and C. Goutte, "A co-classification approach to learning from multilingual corpora," *Machine Learning,* vol. 79, no. 1, pp. 105-121, 2010.

[107] A. Louis and T. Newman, "Summarization of Business-related Tweets: A Concept-based Approach," *Proceedings of the 24th International Conference on Computational Linguistics,* 2012.

[108] T. Snowsill, I. Flaounas, T. D. Bie and N. Cristianini, "Detecting Events in a Million New York Times Articles," *Machine Learning and Knowledge Discovery in Databases (ECML/PKDD), Springer,* 2010.

[109] S. Theul, I. Feinerer and K. Hornik, "A tm Plug-In for Distributed Text Mining in R," *Journal of Statistical Software,* vol. 51, no. 5, pp. 1-31, 2012.

[110] J. Dunietz and D. Gillick, "A New Entity Salience Task with Millions of Training Examples," *Proceedings of the European Association for Computational Linguistics,* 2014.

[111] H. Steck, R. v. Zwol and C. Johnson, "Interactive Recommender Systems: Tutorial," *RecSys '15 Proceedings of the 9th ACM Conference on Recommender Systems ,* pp. 359-360, 2015.

[112] J. Daiber, M. Jakob, C. Hokamp and P. N. Mendes, "Improving Efficiency and Accuracy in Multilingual Entity Extraction," *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics),* 2013.

[113] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *CoRR,* vol. abs/1301.3781, 2013.

[114] J. Garten, K. Sagae, V. Ustun and M. Dehghani, "Combining Distributed Vector Representations for Words," *Proceedings of NAACL-HLT,* pp. 95-101, 2015.

[115] Y. Bengio, "Practical Recommendation for Gradient-Based Training of Deep Architectures," *Neural Networks: Tricks of the Trade,* vol. Neural Networks: Tricks of the Trade: Second Edition, pp. 437-478, 2012.

[116] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research,* vol. 13, pp. 281-305, 2012.

[117] J. Heaton, Introduction to Neural Networks for Java, 2Nd Edition, Heaton Research, Inc., 2008.

[118] Y. Xu, R. Jia, L. Mou, G. Li, Y. Chen, Y. Lu and Z. Jin, "Improved Relation Classification by Deep Recurrent Neural Networks," *abs/1601.03651,* 2016.

[119] J. Y. Lee and F. Dernoncourt, "Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks," *CoRR,* vol. eprint arXiv:1603.03827, 2016.

APPENDIX

A.1 RAW NEW YORK TIMES ARTICLE

```
alternativeURL:          http://www.nytimes.com/2007/01/27/world/middleeast/27policy.html
articleAbstract:         Pres Bush and his senior aides justify US actions ...
authorBiography:         null
banner:                  null
biographicalCategories:  []
body:                    President Bush and his senior aides on Friday justified American ...
byline:                  By MARK MAZZETTI and DAVID S. CLOUD
columnName:              null
columnNumber:            1
correctionDate:          null
correctionText:          null
credit:                  The New York Times
dateline:                WASHINGTON, Jan. 26
dayOfWeek:               Saturday
descriptors:             [Shiite Muslims, United States International Relations, United States Armament and Defense]
featurePage:             null
generalOnlineDescriptors:[United States International Relations, United States International Relations, Shiite Muslims,
                         International Relations, United States Armament and Defense, Armament, Defense and Military
                         Forces, United States Politics and Government, Politics and Government]
guid:                    1821747
headline:                Bush Defends Moving Against Iranians Who Help Shiites Attack U.S.-Led Forces in Iraq
kicker:                  null
leadParagraph:           President Bush and his senior aides on Friday ...
locations:               [Iraq, Iran]
names:                   []
newsDesk:                Foreign Desk
normalizedByline:        Cloud, David S; Mazzetti, Mark
onlineDescriptors:       [United States Armament and Defense]
onlineHeadline:          Bush Defends Moving Against Iranians Who Help Shiites Attack U.S.-Led Forces in Iraq
onlineLeadParagraph:     The president and his senior aides justified ...
onlineLocations:         [Iraq, Iran]
onlineOrganizations:     []
onlinePeople:            []
onlineSection:           World; Washington
onlineTitles:            []
organizations:           []
page:                    8
people:                  [Mazzetti, Mark, Cloud, David S, Bush, George W (Pres)]
publicationDate:         Sat Jan 27 00:00:00 CET 2007
publicationDayOfMonth:   27
publicationMonth:        1
publicationYear:         2007
section:                 A
seriesName:              null
slug:                    27policy
sourceFile:              Z:\test_test\1821747.xml
taxonomicClassifiers:    [Top/News, Top/News/World/Countries and Territories/Iran, Top/News/World/Countries and Territories/
                         Iraq, Top/News/World, Top/News/Washington, Top/News/Washington/Campaign 2004/Candidates, Top/News/
                         World/Middle East, Top/Features/Travel/Guides/Destinations/Middle East, Top/Features/Travel/Guides/
                         Destinations/Middle East/Iran, Top/Features/Travel/Guides/Destinations/Middle East/Iraq]
titles:                  []
typesOfMaterial:         []
url:                     http://query.nytimes.com/gst/fullpage.html?res=9C01EFDB173FF934A15752C0A9619C8B63
wordCount:               1001

descriptors:             [Shiite Muslims, United States International Relations, United States Armament and Defense]
generalOnlineDescriptors:[United States International Relations, United States International Relations, ...
guid:                    1821747
headline:                Bush Defends Moving Against Iranians Who Help Shiites Attack U.S.-Led Forces in Iraq
onlineDescriptors:       [United States Armament and Defense]
taxonomicClassifiers:    [Top/News, Top/News/World/Countries and Territories/Iran, Top/News/World/Countries and Territories/
                         Iraq, Top/News/World, Top/News/Washington, Top/News/Washington/Campaign 2004/Candidates, Top/News/
                         World/Middle East, Top/Features/Travel/Guides/Destinations/Middle East, Top/Features/Travel/Guides/
                         Destinations/Middle East/Iran, Top/Features/Travel/Guides/Destinations/Middle East/Iraq]
url:                     http://query.nytimes.com/gst/fullpage.html?res=9C01EFDB173FF934A15752C0A9619C8B63
```