ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

On Authorship Attribution

Relatore:
Chiar.mo Prof.
DANILO MONTESI

Correlatore:
Dott.
FLAVIO BERTINI

Presentata da:
Gabriele Calarota

Sessione III
Anno Accademico 2019-2020

“When I was in college,
I wanted to be involved in things that would change the world”
Elon Musk

SOMMARIO

ICD (International Classification of Diseases), ovvero la classificazione internazionale delle malattie, è un sistema standard di classificazione ampiamente usato, che codifica un grande numero di specifiche malattie, sintomi, infortuni e procedure mediche in classi numeriche. Assegnare un codice ad un caso clinico significa classificarlo in una o più classi discrete, permettendo studi statistici e procedure di calcolo automatico. E' evidente che la possibilità di avere un codice discreto invece di una frase in linguaggio naturale ha un enorme vantaggio per i sistemi di manipolazione dei dati. L'uso di questo sistema di classificazione, ufficialmente alla decima revisione (ICD-10-CM)[1], diventa sempre più importante per ragioni legate alle polizze assicurative e potrebbe interessare anche i bilanci amministrativi dei reparti ospedalieri.

Ottenere un classificatore automatico accurato è però un arduo compito in quanto la revisione ICD-9-CM conta più di 14 mila classi, quasi 68 mila nella revisione ICD-10-CM. Ottenere un training set soddisfacente per un Classificatore Testuale (TC) è quasi impossibile: sono rari i testi medici ufficiali etichettati con i codici, mentre le diagnosi reali da classificare sono scritte in gergo medico e piene di errori ortografici.

Avendo un training set piuttosto ristretto ci aspettiamo che ampliandolo con un corpus di dati testuali in ambito medico, migliori l'accuratezza del classificatore automatico.

Questo lavoro di tesi descrive innanzitutto come costruire e mettere insieme un dataset con soli dati testuali in ambito medico-specifico. Questo dataset viene, in secondo luogo, manipolato con la tecnica del 'word embedding' (i.e. *immersione di parole*) che associa informazioni semantiche e sintattiche delle parole tramite numeri, costruendo uno spazio vettoriale in cui i vettori delle parole sono più vicini se le parole occorrono negli stessi contesti linguistici, cioè se sono riconosciute come semanticamente più simili.

Viene presentato, infine, come un word embedding di dominio specifico[2] aiuti a migliorare il classificatore automatico, e sia quindi preferibile ad un word embedding di tipo generico.

---

[1] Attualmente non ancora adottata in Italia

[2] In questo caso in ambito medico

ABSTRACT

In this work we evaluate domain-specific embedding models induced from textual resources in the medical domain. The International Classification of Diseases (ICD) is a standard, broadly used classification system, that codes a large number of specific diseases, symptoms, injuries and medical procedures into numerical classes. Assigning a code to a clinical case means classifying that case into one or more particular discrete class, hence allowing further statistics studies and automated calculations. The possibility to have a discrete code instead of a text in natural language is intuitively a great advantage for data processing systems. The use of such classification is becoming increasingly important for, but not limited to, economic and policy-making purposes. Experiments show that domain-specific word embeddings, instead of a general one, improves classifiers in terms of frequency similarities between words.

CONTENTS

1 Introduction **13**

1.0.1 Motivation and Problem Statement 13
1.0.2 Thesis Structure 15

2 Authorship attribution's tasks **17**

2.1 History of methodologies 17
2.2 Method's approach 20
2.2.1 Profile-based approach 20
2.2.2 Instance-based approach 21
2.3 The real problem 21
2.3.1 Profiling problem 22
2.3.2 Needle-in-hay-stack problem 23
2.3.3 Verification problem 24

3 Text characteristics analysis **27**

3.1 Character Features 28
3.1.1 Affix n-grams 28
3.1.2 Word n-grams 29
3.1.3 Punctuation n-grams 29
3.2 Lexical Features 30
3.2.1 Bag of Words 30
3.2.2 Word N-grams 33
3.2.3 Vocabulary Richness 35
3.2.4 Stylometric features 36
3.2.5 Function Words 37
3.2.6 Tf-Idf 38
3.3 Syntactic Features 39
3.4 Semantic Features 40
3.4.1 Positivity and Negativity index 40
3.5 Application Specific Features 42

3.5.1 Vector embeddings of words (Word2Vec) 42
3.5.2 Vector embeddings of paragraphs(Sentence2Vec) 43
3.5.3 Vector embeddings of documents (Doc2Vec) 43

4 State of the art: Data collection and Techniques for Authorship Attribution 45
4.1 SVM studies 46
4.1.1 SVM studies on authorship attribution 46
4.2 GDELT studies 46
4.2.1 Victorian era books 46
4.2.2 Authorship attribution GDELT 46
4.3 RCV1 studies 46
4.3.1 Studies on RCV1 on authorship attribution 46
4.4 The guardian studies 46
4.4.1 Cross-topic authorship attribution 46
4.5 Studies on Stanford Amazon Food Reviews 46
4.6 Dataset selection 46

5 Our approach 61

6 Result and Evaluation 63

7 Future works 65

8 Conclusion 75

Bibliography 77

A Code 81
A.1 Dataset estraction 81
A.1.1 RCV1 81
A.1.2 GDELT 81
A.2 Model 82
A.2.1 Feature extraction 82
A.2.2 Train model 82
A.2.3 Evaluation 82

CHAPTER 1

INTRODUCTION

1.0.1 Motivation and Problem Statement

The International Classification of Diseases (ICD) is a standard, broadly used classification system, that codes a large number of specific diseases, symptoms, injuries and medical procedures into numerical classes. Assigning a code to a clinical case means classifying that case into one or more particular discrete class, hence allowing further statistics studies and automated calculations. The possibility to have a discrete code instead of a text in natural language is intuitively a great advantage for data processing systems. The use of such classification is becoming increasingly important for, but not limited to, economic and policy-making purposes. While the ICD Classification is clearly useful on many aspects, physicians and clinical personnel think and write in natural language and, after that, assign the right code to their text description aided by manuals, guidelines, or their own memory. For this reason, the task is often assigned to health professional trained in medical classification. The ICD-9-CM contains more than 16 thousands classification codes for diseases and ICD-10-CM counts over 68 thousands of diagnosis, meaning that manual methods are inadequate to locate the right classes in a real-world scenario, even for expert clinical coders. In some medical departments the codes used are just a tiny subset of the classification set, hence the problem is reduced, but in many other and in generic departments like the Emergency, this subset covers a big portion of the classification codes. Among the many attempts to simplify or automate the coding task of medical text we can distinguish between two approaches: the Information Retrieval(IR) of codes from a dictionary and the machine learning or rule-based Text Classification (TC). While the first technique is still broadly used in real world applications, due to his simplicity of implementation, over the last years, TC has received attention as a valuable solution to medical text coding.[**?**]

The described problem fall into a text classification problem with some properties:

1. **Multi-class Classification:** the number of output classes(ICD codes) is very high, contrary to the simplest binary classification
2. **Multi-label Classification:** a text instance can be associated with more than one label. This is true for two reasons: because a text can include different disease and because there might need more than one code to describe a clinical condition.

The TC approach to the problem is the most promising one, since it provides automatic code assignment given enough samples data for each code to train the classifier. Unfortunately this last assumption is very hard to satisfy: labeled medical texts are rare and often roughly coded, besides the text to be classified is in a jargon language and filled of typing errors. Even getting a clean and balanced training set of labeled medical text, text classification achieved great results on small datasets, but almost fails in classifying large-scale taxonomies, like the ICD, in both classification accuracy and performance.[**?**] Many past studies indicated that data imbalance problem can severely affect the classifier's performance. For example, (**?** , **?**)[**?**] found that 874 of 1,231 ICD-9-CM codes in UKLarge dataset have less than 350 supporting data, whereas only 92 codes have more than 1,430 supporting data. The former group has macro F1 value of 51.3%, but the latter group only has 16.1%. To resolve data imbalance problem, they used optimal training set (OTS) selection approach to sample negative instance subset that provides best performance on validation set. However, OTS did not work on UKLarge dataset because several codes have so few training examples that even carefully selecting negative instances could not help. We expect that preparing the dataset in a better way, the classifier will respond better in terms of frequency similarities between words. This can be satisfied with word embedding, that is a type of mapping words into numbers that allows words with similar meaning to have similar vectorial representation. As well as being amenable to processing by Machine Learning algorithms, this vector representation has two important and advantageous properties:

1. **Dimensionality Reduction** - it is a more efficient representation
2. **Contextual Similarity** - it is a more expressive representation

If you're familiar with the Bag of Words approach, you'll know it often results in huge, very sparse vectors, where the dimensionality of the vectors representing each document is equal to the size of the supported vocabulary. Word Embedding aims to create a vector representation with a much lower dimensional space. Word Embedding is used for semantic parsing, to extract meaning from text to enable natural language understanding. For a language model to be able to predict the meaning of text, it needs to be aware of the contextual similarity of words. For instance, that we tend to find fruit words (like apple or orange) in sentences where they're grown, picked, eaten and juiced, but wouldn't expect to find those same concepts in such close proximity to, say, the word aeroplane. The vectors created by Word Embedding preserve these similarities, so words that regularly occur nearby in text will also be in close proximity in vector space. So word embeddings means building a low-dimensional vector representation from corpus of text, which preserves the contextual similarity of words. An interesting feature of word vectors is that because they're numerical representations of contextual similarities between words (which might be gender, tense, geography or something else entirely), they can be manipulated arithmetically just like any other vector. [See Figure 7.4]

In this work, we gathered together 3 main corpora of specific medical data to produce a domain-specific word embedding model. The three main dataset are taken from the emergency room discharge records of the Forlì Hospital, where we collected more than 700k real anonymous diagnosis written by doctors when sending home patients. The second main corpus has been downloaded from all the italian medical articles available on Wikipedia and the last one was the official ICD-9-CM dictionary of the more than 16k definitions of diagnosis and their corresponding code, each one different for every possible diagnosis in the corpus.
We trained the datasets joined together forming a domain-specific italian corpus of medical data, producing a domain-specific word embedding model that will be preferred to a general purpose one, when training the classifier. Domain-specific, technical vocabulary presents a challenge to NLP applications. The majority of work dealing with intrinsic evaluation of word embeddings has focused on general domain embeddings and semantic relations between frequent and generic terms. However, it has been shown that embeddings differ from one domain to another due to lexical and semantic variation (**?** , **?** ; **?** , **?**). Domain-specific terms are challenging for general domain embeddings since there are few statistical clues in the underlying corpora for these items (**?** , **?**). In fact, we have found that testing technical word similarities in medical environment between domain-specific model and general purpose, the former responds better. In a related work we built the automatic ICD-9-CM classifier using neural network and weighting words with our word embeddings. For evaluation reasons, we tested both a general purpose word embedding and our model produced, finding out that the accuracy is much better with our domain-specific model.

1.0.2 Thesis Structure

The rest of this thesis is organized into the following chapters:

- **Chapter 2**. Chapter 2 provides a word embedding background, the main topic on which this thesis is based. We will discuss some of the most popular methods among numerical word embeddings and words representation. Then we will explain some of the most recent related work on word embeddings and ICD-9-CM classification.

- **Chapter 3**. Chapter 3 presents the datasets; we divided them in our domain-specific dataset, from where each part of it was taken from and how to reproduce it. At the end of the chapter we present also the general purpose dataset used as a comparison for our less popular domain-specific word embedding.
- **Chapter 4**. Chapter 4 provides the results obtained by showing most similar words in our evaluated models. It shows also characteristics of our models and dataset, with most frequently words, medical jargon, typo errors and comparison between domain-specific models and general domain one. At the end of the chapter we present the results of the F1 calculated by a classifier that used a general purpose word embedding and another one that used our domain-specific word embeddings.

CHAPTER 2

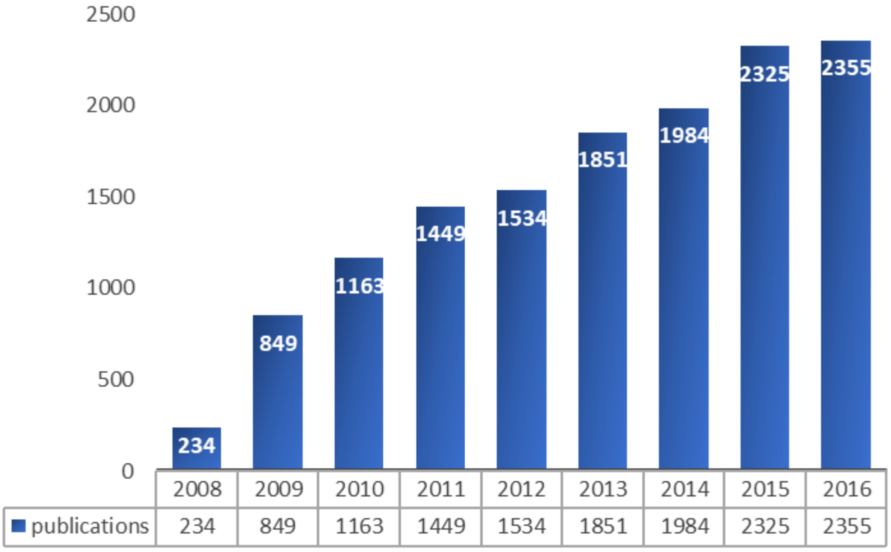
Authorship attribution's tasks

The baseline of the text mining is to obtain little pieces of desired information over tons of text data without having to read everything.

The vast numbers of biomedical text provide a rich source of knowledge for medical research. Text mining can help us to mine information and knowledge from a mountain of text and it is now widely applied. As shown in Figure 7.1, the number of publications obtained from PubMed using '*text mining*' as the query word in the title or abstract has grown substantially since 2000. Many researchers have taken advantage of text mining technology to discover novel knowledge to improve the development of biomedical research. [?]

2.1 History of methodologies

The purpose of Text Mining is to process unstructured (textual) information, extract meaningful numeric indices from the text, and, thus, make the information contained in the text accessible to the various data mining (statistical and machine learning) algorithms. Information can be extracted to derive summaries for the words contained in the documents or to compute summaries for the documents based on the words contained in them. Hence, you can analyze words, clusters of words used in documents, etc., or you could analyze documents and determine similarities between them or how they are related to other variables of interest in the data mining project. In the most general terms, text mining will 'turn text into numbers' (meaningful indices), which can then be incorporated in other analyses such as predictive data mining projects, the application



	2008	2009	2010	2011	2012	2013	2014	2015	2016
■ publications	234	849	1163	1449	1534	1851	1984	2325	2355

Figure 2.1: The number of publications in PubMed using the query word "*text mining*" or "*literature mining*" in the title or abstract over the last years. Search detail: *text mining* [Title/Abstract] or *literature mining* [Title/Abstract].

of unsupervised learning methods (clustering), etc. These methods are described and discussed in great detail in the comprehensive overview work by **?** **?** .

Text mining employs many computational technologies, such as machine learning, natural language processing, biostatistics, information technology, and pattern recognition, to find new exciting outcomes hidden in unstructured biomedical text. The goal of text mining is to derive implicit knowledge that hides in unstructured text and present it in an explicit form. This generally has four phases: information retrieval, information extraction, knowledge discovery, and hypothesis generation. Information retrieval systems aim to get desired text on a certain topic; information extraction systems are used to extract predefined types of information such as relation extraction; knowledge discovery systems help us to extract novel knowledge from text; hypothesis generation systems infer unknown biomedical facts based on text. Thus, the general tasks of biomedical text mining include information retrieval, named entity recognition and relation extraction, knowledge discovery and hypothesis generation. [**?**]

To reiterate, text mining can be summarized as a process of 'numericizing' text. At the simplest level, all words found in the input documents will be indexed and counted in order to compute a table of documents and words, i.e., a matrix of frequencies that enumerates the number of times that each word occurs in each document. This basic process can be further refined to exclude certain common words such as 'the' and 'a' (stop word lists) and to combine different grammatical forms of the same words such as 'traveling', 'traveled', 'travel', etc. (This process is known as stemming[1]). However, once a table of (unique) words (terms) by documents has been derived, all standard statistical and data mining techniques can be applied to derive dimensions or clusters of words or documents, or to identify 'important' words or terms that best predict another outcome variable of interest.

---

[1] The term stemming refers to the reduction of words to their roots so that, for example, different

Once the input documents have been indexed and the initial word frequencies (by document) computed, a number of additional transformations can be performed to summarize and aggregate the information that was extracted. As described above, the most basic result of the initial indexing of words found in the input documents is a frequency table with simple counts, i.e., the number of times that different words occur in each input document. Usually, we would transform those raw counts to indices that better reflect the (relative) 'importance' of words and/or their semantic specificity in the context of the set of input documents (see the discussion of inverse document frequencies, above). A common analytic tool for interpreting the 'meaning' or 'semantic space' described by the words that were extracted, and hence by the documents that were analyzed, is to create a mapping of the word and documents into a common space, computed from the word frequencies or transformed word frequencies (e.g., inverse document frequencies).

After significant (e.g., frequent) words have been extracted from a set of input documents, and/or after singular value decomposition has been applied to extract salient semantic dimensions, typically the next and most important step is to use the extracted information in a data mining project.

- **Graphics (visual data mining methods).** Depending on the purpose of the analyses, in some instances the extraction of semantic dimensions alone can be a useful outcome if it clarifies the underlying structure of what is contained in the input documents. For example, a study of new car owners' comments about their vehicles may uncover the salient dimensions in the minds of those drivers when they think about or consider their automobile (or how they 'feel' about it). For marketing research purposes, that in itself can be a useful and significant result. You can use the graphics (e.g., 2D scatterplots or 3D scatterplots) to help you visualize and identify the semantic space extracted from the input documents.

- **Clustering and factoring.** You can use cluster analysis methods to identify groups of documents (e.g., vehicle owners who described their new cars), to identify groups of similar input texts. This type of analysis also could be extremely useful in the context of market research studies, for example of new car owners. You can also use Factor Analysis and Principal Components and Classification Analysis (to factor analyze words or documents).

---

grammatical forms or declinations of verbs are identified and indexed (counted) as the same word. For example, stemming will ensure that both 'travel' and 'traveled' will be recognized by the program as the same word.

- **Predictive data mining.** Another possibility is to use the raw or transformed word counts as predictor variables in predictive data mining projects.

2.2 Method's approach

In Numeric Natural Language Processing (NLP), we often map words into vectors that contains numeric values so that machine can understand it. Word embedding is a type of mapping that allows words with similar meaning to have similar vectorial representation.

2.2.1 Profile-based approach

A traditional way of representing words is one-hot vector, which is essentially a vector with only one target element being 1 and the others being 0. The length of the vector is equal to the size of the total unique vocabulary in the corpora. Conventionally, these unique words are encoded in alphabetical order. Namely, you should expect the one-hot vectors for words starting with "a" with target "1" of lower index, while those for words beginning with "z" with target "1" of higher index.

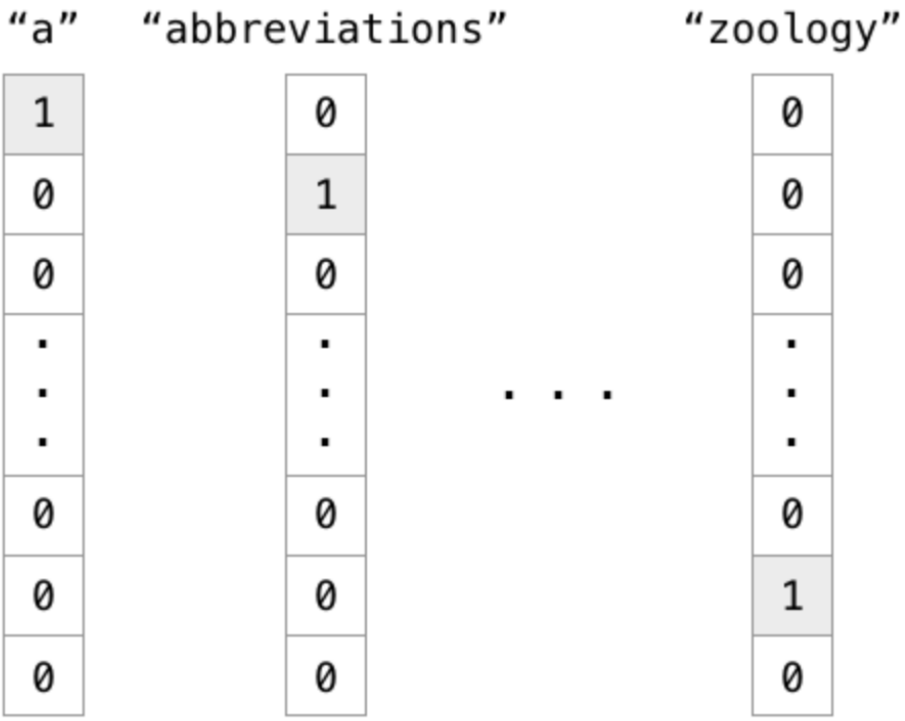


Figure 2.2: One Hot Vector: A simple and easy way to implement word vectors. 1 Bit set to 1 and all the others to 0. The dimension of the vector depends on the size of the vocabulary in input.

Though this representation of words is simple and easy to implement, there are several issues. First, you cannot infer any relationship between two words given their one-hot representation. For instance, the word “endure” and “tolerate”, although have similar meaning, their targets “1” are far from each other. In addition, sparsity is another issue as there are numerous redundant “0” in the vectors. This means that we are wasting a lot of space.We need a better representation of words to solve these issues.

2.2.2 Instance-based approach

Word2Vec[2] is an efficient solution to these problems, which leverages the context of the target words. Essentially, we want to use the surrounding words to represent the target words with a Neural Network whose hidden layer encodes the word representation.

There are two types of Word2Vec, Skip-gram and Continuous Bag of Words (CBOW). I will briefly describe how these two methods work in the following paragraphs.

2.3 The real problem

For skip-gram, the input is the target word, while the outputs are the words surrounding the target words. For instance, in the sentence “I have a cute dog”, the input would be “a”, whereas the output is “I”, “have”, “cute”, and “dog”, assuming the window size is 5. All the input and output data are of the same dimension and one-hot encoded. The network contains 1 hidden layer whose dimension is equal to the embedding size, which is smaller than the input/output vector size. At the end of the output layer, a softmax activation function is applied so that each element of the output vector describes how likely a specific word will appear in the context. In mathematics, the softmax function, or normalized exponential function is a generalization of the logistic function that *squashes* a K-dimensional vector z of arbitrary real values to a K-dimensional vector δ(z) of real values, where each entry is in the range (0,1) and all the entries add up to 1. The target is a (K-1)-dimensional space, so one dimension has been lost. [**?**]

The graph below visualizes the network structure. [Fig 7.3]

With skip-gram, the representation dimension decreases from the vocabulary size (V) to the length of the hidden layer (N). Furthermore, the vectors are more “meaningful” in terms of describing the relationship between words. The vectors obtained by subtracting two related words sometimes express a meaningful concept such as gender or verb tense, as shown in the following figure (dimensionality reduced). [Fig 7.4]

---

[2] `https://code.google.com/archive/p/word2vec/`

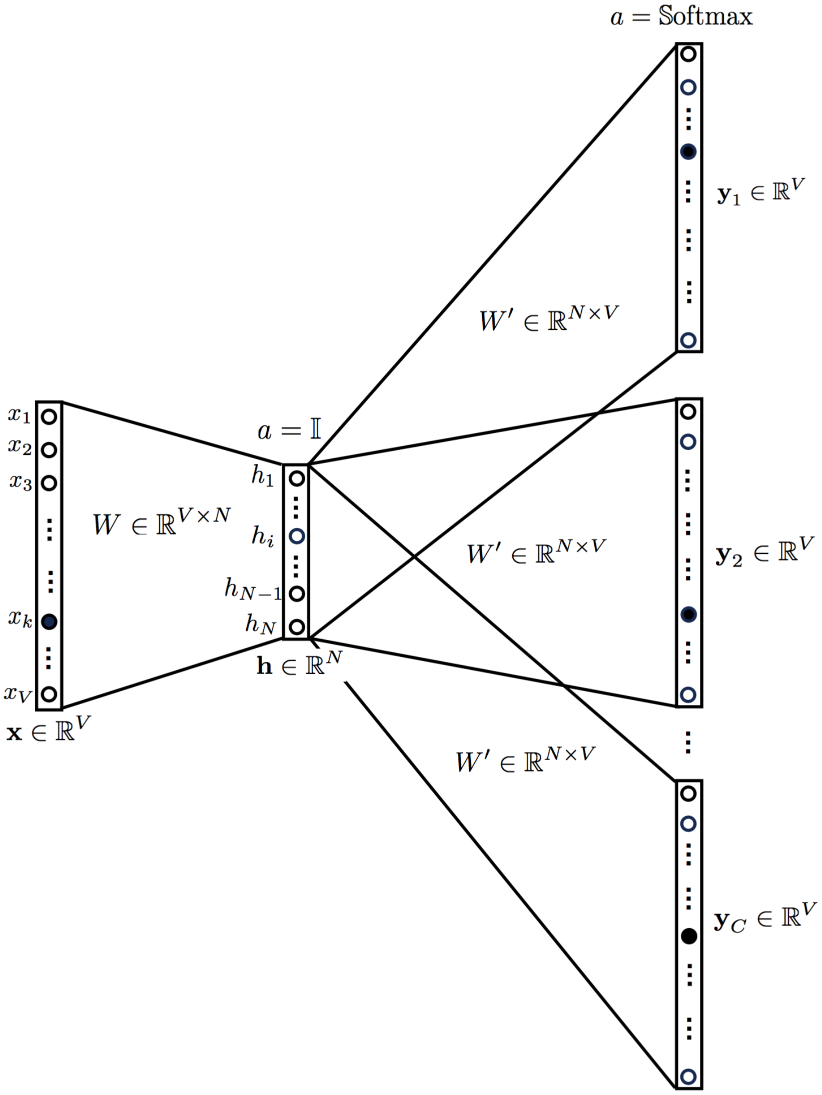

$a = \mathbb{S}\text{oftmax}$
$\mathbf{y}_1 \in \mathbb{R}^V$
$W' \in \mathbb{R}^{N\times V}$
x_1
x_2
x_3
$a = \mathbb{I}$
h_1
$W \in \mathbb{R}^{V\times N}$
h_i
$W' \in \mathbb{R}^{N\times V}$
$\mathbf{y}_2 \in \mathbb{R}^V$
h_{N-1}
x_k
h_N
$\mathbf{h} \in \mathbb{R}^N$
x_V
$\mathbf{x} \in \mathbb{R}^V$
$W' \in \mathbb{R}^{N\times V}$
$\mathbf{y}_C \in \mathbb{R}^V$

Figure 2.3: Skip-gram: The word embedding for the target words can be obtained by extracting hidden layers after feeding the one-hot representation of that word into the network.

2.3.1 Profiling problem

Continuous Bag of Words (CBOW)[3] is very similar to skip-gram, except that it swaps the input and output. The idea is that given a context, we want to know which word is most likely to appear in it.

The biggest difference between Skip-gram and CBOW is that the way the word vectors are generated. For CBOW, all the examples with the target word as target are fed into the networks, and taking the average of the extracted hidden layer. For example, assume we only have two sentences, "He is a nice guy" and "She is a wise queen". To compute the word representation for the word "a", we need to feed in these two examples, "He is nice guy", and "She is wise queen" into the Neural Network and take the average of

---

[3] `https://iksinc.online/tag/continuous-bag-of-words-cbow/`

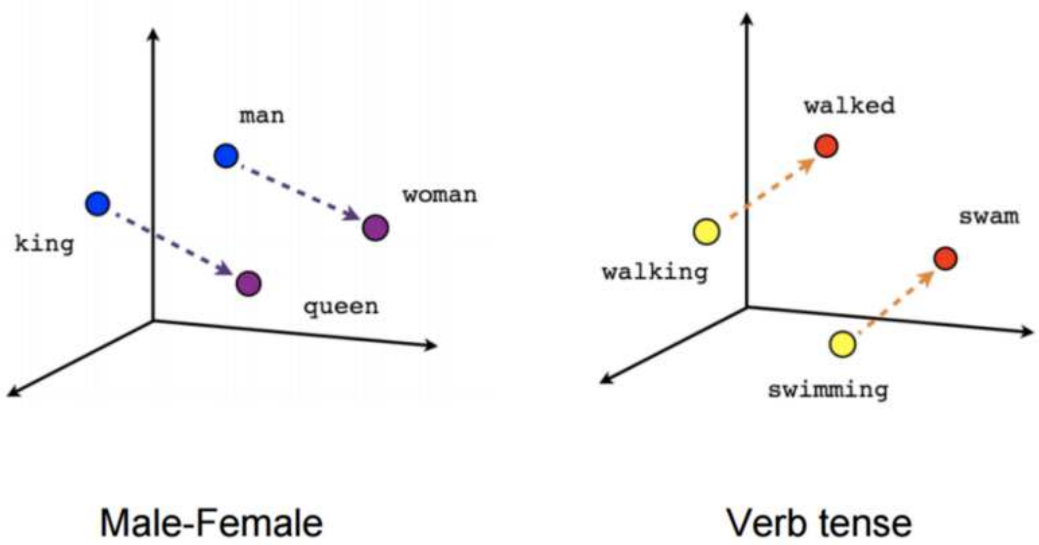


Figure 2.4: Skip-gram: the vectors are more "meaningful" in terms of describing the relationship between words.

the value in the hidden layer. Skip-gram only feed in the one and only one target word one-hot vector as input.

Figure 7.5 shows a 3D representation of word vectors, after a applying a nonlinear dimensionality reduction function like (t-SNE)[4][**?**]. The key to understand here is that having the vectors in 2 or 3 dimensions we can then move in some direction and find terms given by the contest. If we move through the male-female direction from the word vector representing the word *man*, we are likely to find the word *woman*. Likewise we are going to find the word *queen* if we start from the word vector representing the word *queen*.

It is claimed that Skip-gram tends to do better in rare words. Nevertheless, the performance of Skip-gram and CBOW are generally similar.

2.3.2 Needle-in-hay-stack problem

FastText is an extension to Word2Vec proposed by Facebook in 2016. Instead of feeding individual words into the Neural Network, FastText breaks words into several n-grams (sub-words). For instance, the tri-grams for the word apple is app, ppl, and ple (ignoring the starting and ending of boundaries of words). The word embedding vector for apple will be the sum of all these n-grams. After training the Neural Network, we will have word embeddings for all the n-grams given the training dataset. Rare words can now be properly represented since it is highly likely that some of their n-grams also appears in other words.[**?**]

---

[4] t-distributed Stochastic Neighbor Embedding

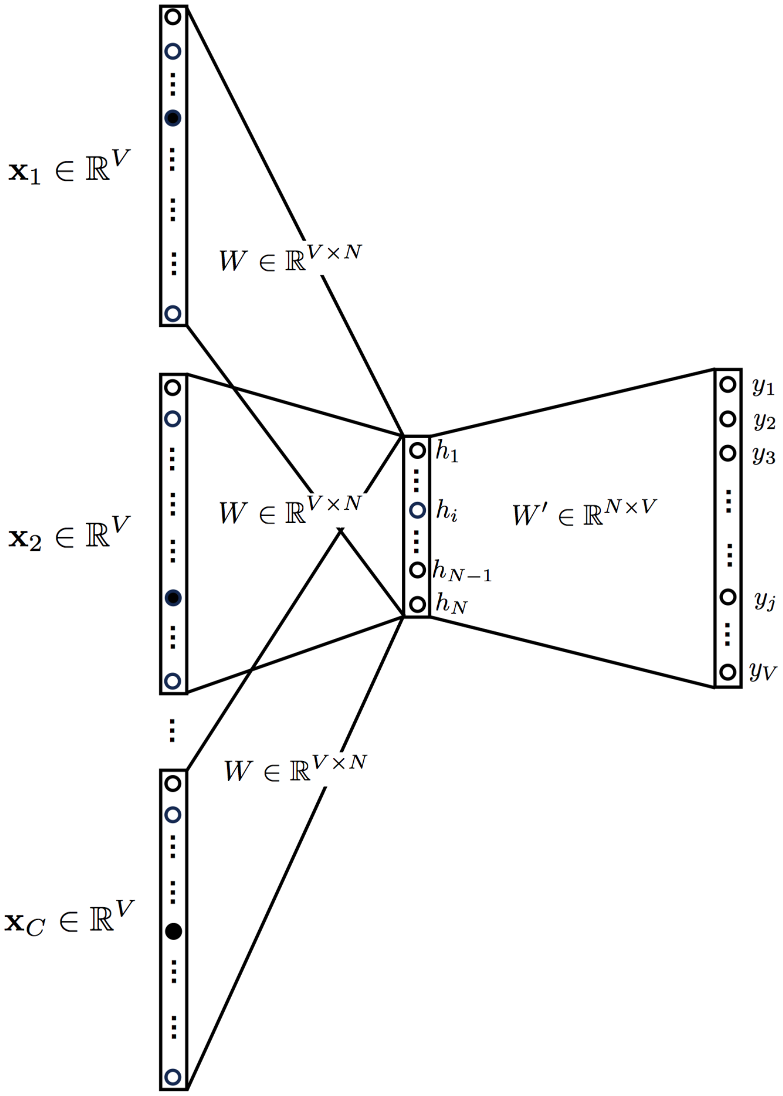

$\mathbf{x}_1 \in \mathbb{R}^V$
$W \in \mathbb{R}^{V \times N}$
$\mathbf{x}_2 \in \mathbb{R}^V$
$W \in \mathbb{R}^{V \times N}$
$\mathbf{x}_C \in \mathbb{R}^V$
$W \in \mathbb{R}^{V \times N}$
h_1
h_i
h_{N-1}
h_N
$W' \in \mathbb{R}^{N \times V}$
y_1
y_2
y_3
y_j
y_V

Figure 2.5: The main difference between CBOW and Skip-gram is that CBOW swaps the input and output.

2.3.3 Verification problem

Due to the success of word embeddings in a variety of NLP applications, some existing studies evaluate word embeddings in representing word semantics quantitatively. Most of them focus on evaluating the word embeddings generated by different approaches. **?** [**?**] presented the first systematic evaluation of word embeddings generated by four models, i.e., DISSECT, CBOW using word2vec, Distributional Memory model, and Collobert and Weston model using a corpus of 2.8 billion tokens in the general English domain. They tested these models on fourteen benchmark datasets in five categories, including semantic relatedness, synonym detection, concept categorization, selectional preferences, and analogy. They found that the word2vec model, CBOW, performed the best for almost all the tasks. **?** [**?**] trained the CBOW model of word2vec, C&W embeddings

[?] , Hellinger PCA [?], GloVe [?], TSCCA [?], and Sparse Random Projections [?] on a 2008 GloVe dump, and tested on the same fourteen datasets. They found that the CBOW outperformed other embeddings on 10 datasets. They also conducted an extrinsic evaluation by using the embeddings as input features to two downstream tasks, namely noun phrase chunking and sentiment classification. They found the results of CBOW were also among the best. **?** [?] conducted a similar intrinsic evaluation, they additionally evaluated the skip-gram models of word2vec, CSLM word embeddings [?], dependency-based word embeddings, and combined word embeddings on four NLP tasks, including Part-Of-Speech tagging, chunking, named entity recognition, mention detection, and two linguistic tasks. They trained these word embeddings on the Gigaword corpus composed of 4 billion words and found that the dependency-based word embeddings gave the best performance on the NLP tasks and that the combination of embeddings yielded significant improvement. However, few of these studies evaluated word embeddings for tasks in the biomedical domain. As most of the aforementioned studies evaluate word embeddings in the general (i.e., non-biomedical) NLP domain, only one recent study by **?** [?] evaluates word embeddings in the biomedical domain, to the best of our knowledge. They trained the CBOW model on two biomedical corpora, namely clinical notes and biomedical publications, and one general English corpora, namely GloVe. The word embeddings were evaluated on subsets of UMNSRS dataset, which consisted of pairs of medical terms with the similarity of each pair assessed by medical experts, and on a document retrieval task and a word sense disambiguation task. They found that the semantics captured by the embeddings computed from biomedical publications were on par with that from clinical notes.

There is a rich body of work on learning general-purpose word embeddings. So far, there are only very few studies focusing on learning domain-specific word embeddings. For example, **?** **?** [?] uses information from a disease lexicon to generate disease-specific word embeddings. The main objective is to bring in-domain words close to each other in the embedding space while pushing out-domain words away from in-domain words. **?** **?** [?] only concerns whether a word is in-domain or not. Another example is **?** **?** [?], describes a novel method to train domain-specific word embeddings from sparse texts. First, it proposes a general framework to encode diverse types of domain knowledge as text annotations; then, it develops a novel Word Annotation Embedding (WAE) algorithm to incorporate diverse types of text annotations in word embedding. Evaluating the method on two text corpora resulted in demonstrating the effectiveness of the method in learning domain-specific word embeddings.

The existing studies of automating ICD-9-CM code assignment can be classified into two groups. Through examining how professional coders assigning ICD-9-CM codes, the first one used rule-based approaches. **?** **?** [?] developed a rulebased system considering factors such as uncertainty, negation, synonymy, and lexical elements. ? ? [?] used Decision Tree (DT) and Maximum Entropy (ME) to automatically generate a rule-based coding system. ? ? [?] composed a hybrid system consisting of a machine learning system with natural language features, a rule-based system based on the overlap between the reports and code descriptions, and an automatic policy system. Their results showed better performance than each single system. The second group employed supervised machine learning methods for the assignment task, and their performance has been being equivalent or even better than those rule-based systems that need experts manually crafting knowledge. ? ? [?] used a stacked model to combine the results of four modules: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Pattern Matching (PM) and a hybrid Medical Text Indexer (MTI) system. ? ? [?] used ME and SVM classifiers, enhanced by a feature engineering module that explores the best combination of several types of features. ? ? [?] proposed a hierarchical text categorization method utilizing the ICD-9-CM codes structure. Along with the introduction of supervised methods, many past studies indicated that data imbalance problem can severely affect the classifier's performance. For example, ? ? [?] found that 874 of 1,231 ICD-9-CM codes in UKLarge dataset have less than 350 supporting data, whereas only 92 codes have more than 1,430 supporting data. The former group has macro F1 value of 51.3%, but the latter group only has 16.1%. To resolve data imbalance problem, they used optimal training set (OTS) selection approach to sample negative instance subset that provides best performance on validation set. However, OTS did not work on UKLarge dataset because several codes have so few training examples that even carefully selecting negative instances could not help. When ? ? [?] found that 85% of the whole death certificate dataset is associated with only top 20 common cancers, whereas the other 65 rarer cancers only have the rest 15% of the dataset, they tried to construct the balanced training set by randomly sampling a static number of negative examples for each class. Their results reflected the benefits of having more training data in improving the classifiers' performance. Since result of original model learned with imbalanced data is not provided, we cannot know the actual improvement. In addition, to deal with codes that only appear once in the dataset, ? ? [?] used a rule-based module to supplement ME and SVM classifiers. For the recent studies, ? ? [?] proposed an automatic feature extraction method, capturing semantic relational tuples. They proved the semantic relational tuple is able to capture information at semantic level and it contribute to ICD-9-CM classification task in two aspects, negation identification and feature generation. Another recent study to solve training data shortage problem, ? ? [?] proposed to strategically draw data from PubMed to enrich the training data when there is such need. The evaluation results indicate that their method can significantly improve the code assignment classifiers' performance at the macro-averaging level.

CHAPTER 3

TEXT CHARACTERISTICS ANALYSIS

The main problem in working with language processing is that machine learning algorithms cannot work on the raw text directly. So, we need some feature extraction techniques to convert text into a matrix(or vector) of features.

In order to identify the authorship of an unknown text document using machine learning the document needs to be quantified first. The simple and natural way to characterize a document is to consider it as a sequence of tokens grouped into sentences where each token can be one of the three: word, number, punctuation mark. To quantify the overall writing style of an author, stylometric features are defined and studied in different domains. Mainly, computations of stylometric features can be categorized into five groups as lexical, character, semantic, syntactic, and application specific features. Lexical and character features mainly considers a text document as a sequence of word tokens or characters, respectively. This makes it easier to do computations comparing to other features. On the other hand, syntactic and semantic features require deeper linguistic analysis and more computation time. Application specific features are defined based on the text domains or languages. These five features are studied and the methods to extract them are also provided for interested readers. Moreover there's a sixth characteristic regarding only hand-written text, which was used for years in the past and it's still studied nowadays [11], which is the *graphological analysis.* Although the problem of recognition of handwriting text is still far from its final solution, in this work, we will focus only on the first 5 characteristics because the main focus since digitalization era has been on studies of digital text characteristics analysis.

3.1 Character Features

Based on these features a sentence consists of a characters sequence. Some of the character-level features are alphabetic characters count, digit characters count, uppercase and lowercase character counts, letter and character n-gram frequencies. This type of feature extraction techniques has been found quite useful to quantify the writing style.[5] A more practical approach in character-level features are the extraction of n-gram characters. This procedure of extracting such features are language independent and requires less complex toolboxes. On the other hand, comparing to word n-grams approach the dimensional of these approaches are vastly increased and it has a curse of dimensional problem. A simple way of explaining what a character n-grams could be with the following example: assume that a word “thesis” is going to be represented by 2-gram characters. So, the resulting sets of points will be “th”, ”he”, ”es”, ”si”, ”is”.
A simple python algorithm is shown in 3.1:

```
def get_char_n_gram(text, n=2):
    """Convert text into character n-grams list"""
    return [text[i:i+n] for i in range(len(text)-n+1)]

# Examples character 2-grams

print(get_char_n_gram("thesis"))
>>Out: [’th’, ’he’, ’es’, ’si’, ’is’]

print(get_char_n_gram("student", n=3))
>>Out: [’stu’, ’tud’, ’ude’, ’den’, ’ent’]
```

Code Listing 3.1: Split word into character n-grams, parametric on n

In [12] have been identified 10 character n-grams categories, being proven as the most successful feature in both single-domain and cross-domain Authorship Attribution.
This 10 categories are grouped into 3 groups: Affix n-grams, Word n-grams, Punctuation n-grams.

3.1.1 Affix n-grams

Character n-grams are generally too short to represent any deep syntax, but some of them can reflect morphology to some degree. In particular, the following affix-like features are extracted by looking at n-grams that begin or end a word:

- **prefix**: A character n-gram that covers the first n characters of a word that is at least n+1 characters long.

- **suffix**: A character n-gram that covers the last n characters of a word that is at least n + 1 characters long.
- **space-prefix**: A character n-gram that begins with a space.
- **space-suffix**: A character n-gram that ends with a space.

3.1.2 Word n-grams

While character n-grams are often too short to capture entire words, some types can capture partial words and other word-relevant tokens. There's a distinction among:

- **whole-word**: A character n-gram that covers all characters of a word that is exactly n characters long.
- **mid-word**: A character n-gram that covers n characters of a word that is at least n + 2 characters long, and that covers neither the first nor the last character of the word.
- **multi-word**: N -grams that span multiple words, identified by the presence of a space in the middle of the n-gram.

3.1.3 Punctuation n-grams

The main stylistic choices that character n-grams can capture are the author's preferences for particular patterns of punctuation. The following features characterize punctuation by its location in the n-gram.

- **beg-punct**: A character n-gram whose first character is punctuation, but middle characters are not.
- **mid-punct**: A character n-gram with at least one punctuation character that is neither the first nor the last character.
- **end-punct**: A character n-gram whose last character is punctuation, but middle characters are not.

In [12] they've observed that in their data almost 80% of the n-grams in the punct-beg and punct-mid categories contain a space. They stated that "this tight coupling of punctuation and spaces is due to the rules of English orthography: most punctuation marks require a space following them". The 20% of n-grams that have punctuation but no spaces correspond mostly to the exceptions to this rule: quotation marks, mid-word hyphens, etc. They've conducted an experiment on RCV1 dataset both the *CCAT_10* *split* and the *CCAT_50 split*. They've also used the Guardian dataset as a cross-domain authorship attribution experiment. In their work they stated that for the single-domain the top four categories for authorship attribution are: *prefix*, *suffix*, *space-prefix* and *mid-word*. On the other hand, for cross-domain authorship attribution the top four categories have been proven to be: *prefix*, *space-prefix*, *beg-punct* and *mid-punct*. For both single-domain and cross-domain authorship attribution, *prefix* and *space-prefix* are strong features, and are generally better than the *suffix* features, perhaps because authors have more control over prefixes in English, while suffixes are often obligatory for grammatical reasons. For cross-domain authorship attribution, *beg-punct* and *mid-punct* they found to be the top features, likely because an author's use of punctuation is consistent even when the topic changes. For single-domain authorship attribution, *mid-word* was also a good feature, probably because it captured lexical information that correlates with authors' preferences towards writing about specific topic.

3.2 Lexical Features

Lexical features relate to the words or vocabulary of a language. It is the very plain way of representing a sentence structure that consists of words, numbers, punctuation marks. These features are very first attempts to attribute authorship in earlier studies [4], [1], [14]. The main advantage of Lexical features is that it is universal and can be applied to any language easily. These features consist of bag of words representation, word N-grams, vocabulary richness, number of punctuation marks, average number of words in a sentence, and many more. Even though the number of lexical features can vary a lot, not all of them are good for every authorship attribution problem. That is why, it is important to know how to extract these features and try out different combinations on different classifiers.

3.2.1 Bag of Words

It is the representation of a sentence with frequency of words. It is a simple and efficient solution but it disregards word-order information. At the very beginning, we applied this representation to our datasets, using the already implemented $CountVectorizer$ from $sklearn.feature_extraction.text$. We gave this hyper-parameter to the function:

- **max_df**=0.8
- **max_features**=10000
- **min_df**=0.02
- **ngram_range**=(1, 2)

In the approach, for each text fragment the number of instances of each unique word is found to create a vector representation of word counts. We capped the max number of features to 10'000 words and discarding the words with a higher frequency of 0.8 across the document and a lower frequency of 0.02. We've also taken into account both single word and word bi-grams. In order to give the reader a better perspective of the impact of this process for the task of authorship attribution, we choose two among the top ten most prolific authors in the RCV1 dataset: *David Lawder* and *Alexander Smith*. In 3.1 we can see the number of documents written by the two selected authors in the RCV1 corpus for the *CCAT* topic.

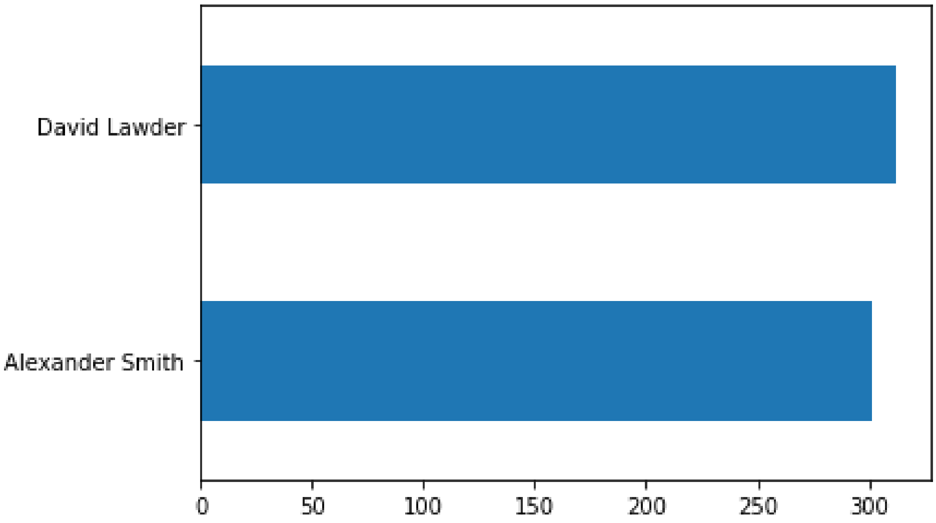


Figure 3.1: The number of documents written by David Lawder and Alexander Smith in the Reuters Corpus for the *CCAT* topic.

With a simple snippet of python code shown in 3.2, we can get the most common words for an author across all the documents we gathered in the dataset.

```
from collections import Counter

def get_most_common_words_in_df(df, n=20):
  """Split a collection of document in single word and order by
                                frequencies across all documents"""
  most_common_words = Counter(" ".join(df["articles"]).split()).
                                most_common(n)
  return most_common_words
```

```
# Examples

# 1. Top 20 words with their frequency for every document written by
                                David Lawder
```

```
print(get_most_common_words_in_df(dataset[dataset['author']=='David
                                          Lawder']))
>>Out: [('the', 7844), ('to', 5133), ('of', 3875), ('and', 3746), ('a
                                          ', 3719), ('in', 3552), ('said',
                                          1749), ('that', 1720), ('for', 1574
                                          ), ('GM', 1453), ('on', 1286), ('at
                                          ', 1267), ('its', 1153), ('The',
                                          1098), ('with', 990), ('is', 957),
                                          ('it', 897), ('will', 875), ('by',
                                          868), ('from', 824)]

# 2. Top 20 words without their frequency for every document written
                                          by David Lawder

print([m[0] for m in get_most_common_words_in_df(dataset[dataset['
                                          author']=='David Lawder'])])
>>Out: ['the',
     'to',
     'of',
     'and',
     'a',
     'in',
     'said',
     'that',
     'for',
     'GM',
     'on',
     'at',
     'its',
     'The',
     'with',
     'is',
     'it',
     'will',
     'by',
     'from']
```

Code Listing 3.2: Top 20 most common words in a document or a collection of document by the same author

As expected top words are determiners that every writer use while constructing an English sentence. For example, for *David Lawder* top 20 words are *"the, to, of, and, a, in, said, that, for, GM, on, at, its, The, with, is, it, will, by, from"* but for *Alexander Smith* they are *"the, of, to, and, a, in, said, was, for, on, it, had, by, be, its, is, with, would, that, as"* in decreasing order. Even though the two sets are mostly the same, the

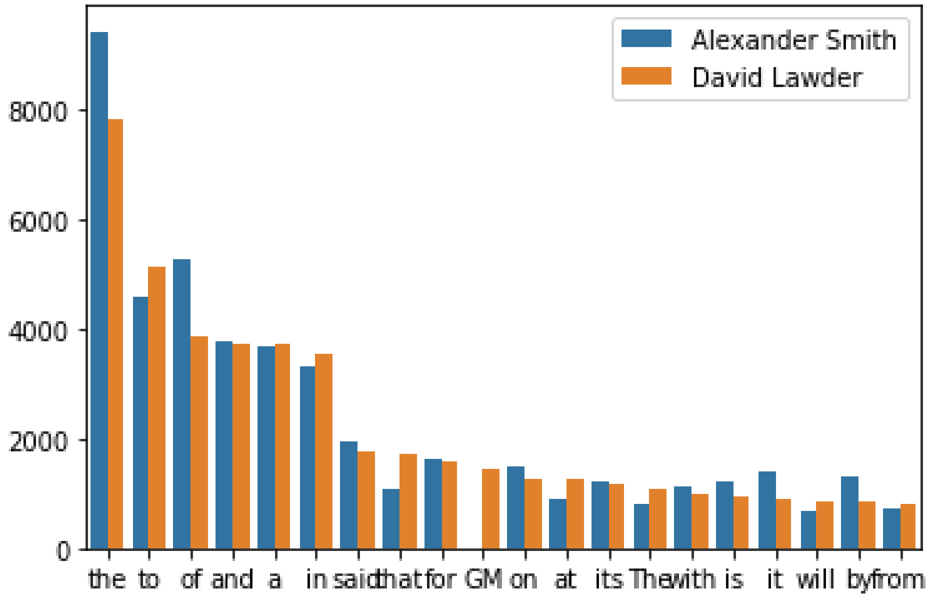


Figure 3.2: Frequency usage of the top 20 words used in the RCV1 corpus by David Lawder, compared to the frequencies of the same words in the corpus by Alexander Smith

orders and the frequency are different for most authors.

The main assumption with authorship attribution problems is that every authors word usage and content differs and based on these differences the work of one author can be differentiated from the other. In order to illustrate this assumption, in 3.2 we can see the top 20 words in the RCV1 corpus in the document written by David Lawder, compared to the same words in the documents of Alexander Smith.

In figure 3.3 & 3.4 we plotted using the Word Cloud identikit of David Lawder and Alexander Smith, generated by every document written by them in the RCV1 corpus.

3.2.2 Word N-grams

It is a type of probabilistic language model for predicting the next item in the form of a (n-1) order. Considering n-grams are useful since Bag of words miss out the word order when considering a text. For example, a verb “take on” can be missed out by Bag of words representation which considers “take” and “on” as two separate words. N-gram also establishes the approach of “Skip-gram” language model. An N-gram is a consecutive sub-sequence of length N of some sequence of sentence while a Skip-gram is a N-length sub-sequence where the components occur at a distance of at most k from each other [9]. In order to extract N-grams from a given text data a simple snippet of code is shown in 3.3, tested for the word grams on the documents written by David Lawder and Alexander Smith of the RCV1 corpus data. No pre-processing on the dataset, such as discarding stopwords, has been done while constructing the N-grams. For David Lawder “the, of the, General Motors Corp.” are the most occurrent uni-gram, bi-gram and three-gram

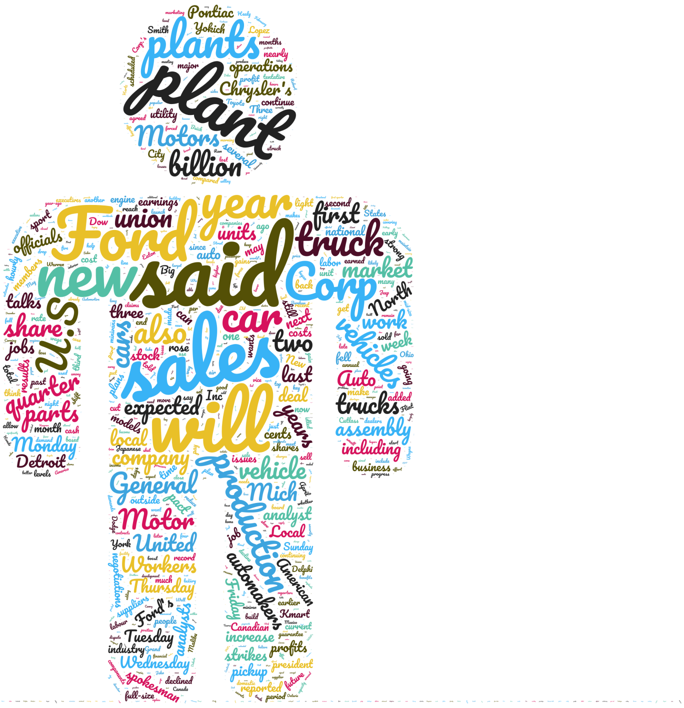

plant
plants
Pontiac
Yokich
Chrysler's
operations
Motors
billion
said
new
Ford
year
truck
Corp
union
first
market
sales
car
U.S.
also
will
share
quarter
parts
Monday
Detroit
expected
company
General
Motor
United
Workers
Thursday
Tuesday
Wednesday
spokesman
production
automakers
Mich
vehicle
vehicles
trucks
assembly
including
Auto
North
work
two
last
deal
years
analyst
Local
American
Friday
Kmart
pickup
strikes
profits
reported

Figure 3.3: Image Generated on for every word in RCV1 corpus data for the documents written by David Lawder

respectively whilst in Alexander Smith documents they are “the, of the, said it would”.

Code Listing 3.3: Get the top 3 most common grams in the corpus, for uni-grams, bi-grams and three-grams

```
from collections import Counter
from nltk import ngrams

def get_Xigram(text, n):
  """Get n-grams for a given text. The number of grams are controlled
                                  by parameter n"""
  return list(ngrams(text.split(), n))

def get_top_3_gram(df):
  """Return a list of three elements, each one containing the top 3
                                 most common grams in the corpus
                                 given as a Dataframe parameter. The
                                  three elements corresponds to the
                                 uni-gram, bi-grams and three-grams.
                                 """
  result = []
  for i in range(1,4):
    result.append(Counter(get_Xigram(" ".join(df["articles"]), i)).
                                 most_common(3))
  return result
```

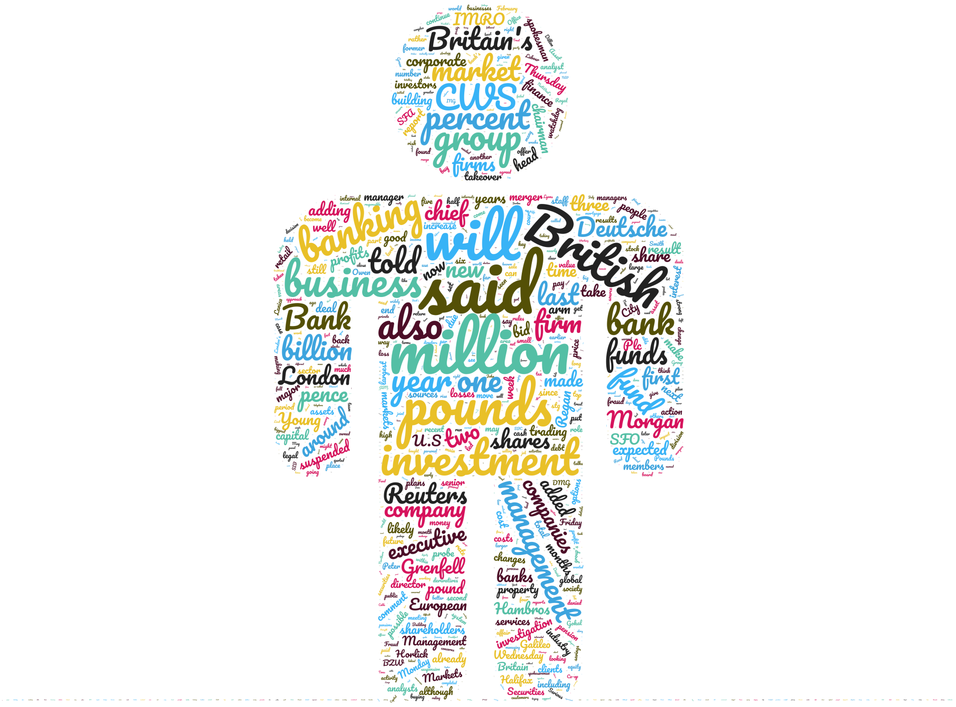

Britain's
market
CWS
percent
group
firms
banking
will
British
said
chief
Deutsche
business
told
new
Bank
also
million
firm
bank
billion
funds
London
year
one
fund
Morgan
pounds
two
shares
investment
Reuters
company
management
companies
executive
Grenfell
European
Hambros

Figure 3.4: Image Generated on for every word in RCV1 corpus data for the documents written by Alexander Smith

```
print(get_top_3_gram(df_david_lawder))
print(get_top_3_gram(df_alexander_smith))
```

Table 3.1: Top 3 uni-grams, bi-grams and three-grams by David Lawder and Alexander Smith in the RCV1 corpus data

Author	Uni-gram	Bi-gram	Three-gram
David Lawder	the	of the	General Motors Corp.
David Lawder	to	in the	United Auto Workers
David Lawder	of	for the	Ford Motor Co.
Alexander Smith	the	of the	said it would
Alexander Smith	of	in the	the end of
Alexander Smith	to	said the	a result of

Table 3.1 contains top 3 uni-grams, bi-grams, three-grams from the authors David Lawder and Alexander Smith.

3.2.3 Vocabulary Richness

It is also referred as vocabulary diversity. It attempts to quantify the diversity of the vocabulary text. It is simply the ratio of V/N where V refers to the total number of unique tokens and N refers to the total number of tokens in the considered texts [13]. As an example, we applied this feature to the RCV1 CCAT_10 dataset[1]. A snippet of the code to extract such feature, is shown in 3.4.

Code Listing 3.4: Calculate vocabulary richness in RCV1 CCAT10 dataset

```
tmp_df = dataset
tmp_df["num_unique_words"] = tmp_df["articles"].apply(lambda x: len
                                (set(str(x).split()))/len(str(x).
                                split()))

objects = {}
for author_i in tmp_df.author.unique():
  objects[author_i] = sum(tmp_df[tmp_df.author==author_i]['
                                num_unique_words'])/len(tmp_df[
                                tmp_df.author==author_i])

plt.bar(range(len(objects)), list(objects.values()), align='center'
                                )
plt.xticks(range(len(objects)), list(objects.keys()))
ax = plt.gca()
plt.setp(ax.get_xticklabels(), rotation=30, horizontalalignment='
                                right')
plt.show()
```

For this portion of the dataset, has been found the lowest vocabulary richness in *Marcel Michelson* and the highest in *Jim Gilchrist*. Overall distribution of vocabulary richness is plotted in Figure 3.5.

3.2.4 Stylometric features

These are features such as number of sentences in a text piece, number of words in a text piece, average number of words in a sentence, average word length in a text piece, number of periods, number of exclamation marks, number of commas, number of colons, number of semicolons, number of incomplete sentences, number of uppercase, title case, camel case, lower case letters. We computed these features comparing the stylometric differences for the documents belonging to David Lawder and the ones of Alexander Smith in the RCV1 corpus. Overall distribution of some of the features introduced here (such as: *number of punctuation, number of words upper case, number of words title, average length of the word, number of stopwords*) are applied and the resulting density measures are calculated for each of the two authors and shown in Table 3.2. Among these

---

[1] The top ten most prolific authors in the RCV1 corpus, selecting the documents belonging to the *CCAT* topic

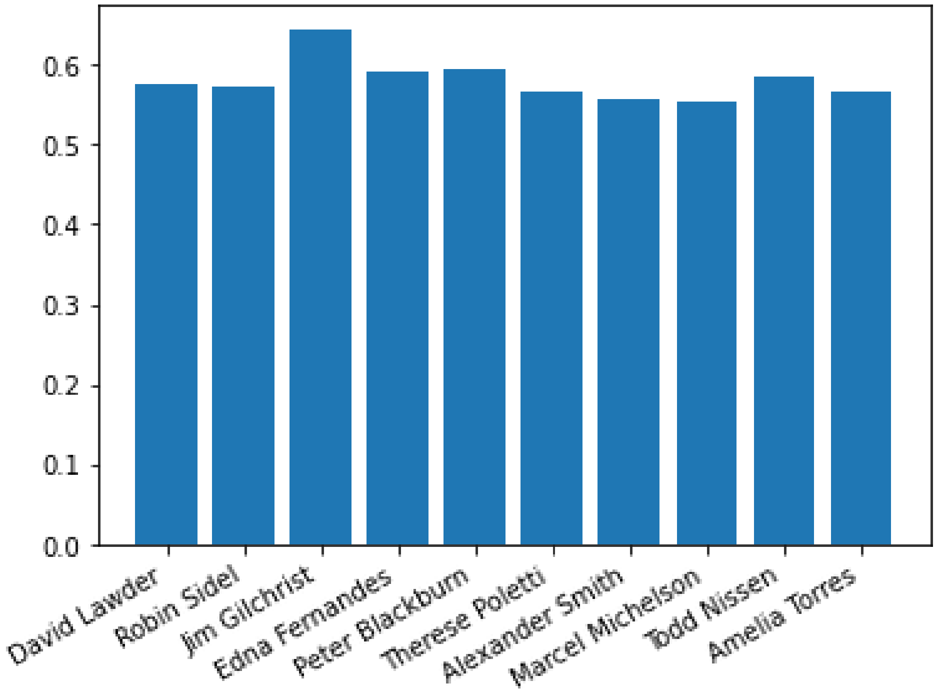


Figure 3.5: Bar Plot of vocabulary richness of RCV1 CCAT_10 authors across all their documents

five features introduced, number of punctuations and number of stop words usage varies the most among the authors and hence they can be better distinguisher comparing to other feature sets.

Table 3.2: Comparing some stylometric features between David Lawder and Alexander Smith calculated on the documents in the RCV1 corpus data and normalized by the number of documents

Stylometric Feature	David Lawder	Alexander Smith
Number of punctuation symbols	101.78	88.73
Number of uppercase words	12.59	9.89
Number of title words	76.59	68.30
Word length mean	5.09	5.11
Number of stopwords	191.38	234.97

3.2.5 Function Words

Function words are the words that have little meaning on their own but they're necessary to construct a sentence in English language. They express grammatical relationships among other words within a sentence, or specify the attitude or mood of the speaker. Some of the examples of function words might be prepositions, pronouns, auxiliary verbs, conjunctions, grammatical articles. Words that are not functions words are called as content words and they can also be studied to further analysis the use case in the authorship attribution problems. The search for a single invariant measure of textual style was natural in the early stages of stylometric analysis, but with the development of more sophisticated multivariate analysis techniques, larger sets of features could be considered. Among the earliest studies to use multivariate approaches was that of Mosteller and Wallace (1964), who considered distributions of FWs. The reason for using FWs in preference to others is that we do not expect their frequencies to vary greatly with the topic of the text, and hence, we may hope to recognize texts by the same author on different topics. It also is unlikely that the frequency of FW use can be consciously controlled, so one may hope that use of FWs for attribution will minimize the risk of being deceived [3]. Many studies since that of Mosteller and Wallace (1964) have shown the efficacy of FWs for authorship attribution in different scenarios [1], [2], [6], [17], confirming the hypothesis that different authors tend to have different characteristic patterns of FW use. Typical modern studies using FWs in English use lists of a few hundred words, including pronouns, prepositions, auxiliary and modal verbs, conjunctions, and determiners. Numbers and interjections are usually included as well since they are essentially independent of topic, although they are not, strictly speaking, FWs. Results of different studies using somewhat different lists of FW have been similar, indicating that the precise choice of FW is not crucial. Discriminators built from FW frequencies often perform at levels competitive with those constructed from more complex features.

3.2.6 Tf-Idf

It stands for term frequency-inverse document frequency. It is often used as a weight in feature extraction techniques. The reason why Tf-Idf is a good feature can be explained in an example. Let's assume that a text summarization needs to be done using few keywords. One strategy is to pick the most frequently occurring terms meaning words that have high term frequency (*tf*). The problem here is that, the most frequent word is a less useful metric since some words like 'a', 'the' occur very frequently across all documents. Hence, a measure of how unique a word across all text documents needs to be measured as well (*idf*). Hence, the product of *tf* x *idf* (3.3) of a word gives a measure of how frequent this word is in the document multiplied by how unique the word is with respect to the entire corpus of documents. Words with a high tf-idf score are assumed to provide the most information about that specific text [13].

$$TF(t) = \frac{\text{Number of times term t appears in a document}}{\text{Total numbers of terms in the document}} \tag{3.1}$$

$$IDF(t) = log_e(\frac{\text{Total number of documents}}{\text{Number of documents with term t in it}}) \tag{3.2}$$

$$Tf - Idf = TF(t) * IDF(t) \tag{3.3}$$

As an example, we buil a Tf-Idf model by considering documents alone within the text corpus for the authors David Lawder and Alexander Smith. In the model, not only the single forms of word tokens but their n-grams are considered as well. Table 3.3 provides the top 5 words with highest Tf-Idf scores for the two authors. Comparing between Table 3.1 and Table 3.3 new meaningful words have appeared that could serve as a new feature for each author such as "dow, kmart, coupe" for David Lawder or "hsbc, pensions, panel" for Alexander Smith.

Table 3.3: Top 5 TFIDF words n-grams by David Lawder and Alexander Smith in the RCV1 corpus data

Author	Token	Value	Author	Token	Value
David Lawder	**dow**	**0.702**	Alexander Smith	**hsbc**	**0.697**
David Lawder	kmart	0.658	Alexander Smith	pensions	0.601
David Lawder	south	0.559	Alexander Smith	bzw	0.592
David Lawder	coupe	0.539	Alexander Smith	panel	0.579
David Lawder	bags	0.517	Alexander Smith	read	0.570

3.3 Syntactic Features

For certain text grammatical and syntactic features could be more useful compared to lexical or character level features. However, this kind of feature extraction techniques requires specific usage of Part of Speech taggers. Some of these features consider the frequency of nouns, adjectives, verbs, adverbs, prepositions, and tense information (past tense,etc). The motivation for extracting these features is that authors tend to use similar syntactic patterns unconsciously [13]. Some researchers are also interested in exploring different dialects of the same language and building classifiers based on features derived from syntactic characteristic of the text. One great example is the work that aims to discriminate between texts written in either the Netherlandic or the Flemish variant of the Dutch language [15]. The feature set in this case consists of lexical, syntactic and word-n grams build on different classifiers and F1-score has been recorded for each cases. Employed syntactic features are function words ratio, descriptive words to nominal words ratio personal pronouns ratio, question words ratio, question mark ratio, exclamation mark ratio [15].

3.4 Semantic Features

Features that we discussed so far aim at analyzing the structural concept of a text such. Semantic feature extraction from text data is a bit challenging. That might explain why there is limited work in this area. One example is the work of Yang who has proposed combination of lexical and semantic features for short text classification [16]. Their approach consists of choosing a broader domain related to target categories and then applying topic models such as *Latent Dirichlet Allocation* to learn a certain number of topics from longer documents. The most discriminative feature words of short text are then mapped to corresponding topics in longer documents [16]. Their experimental results show significant improvements compared to other related techniques studying short text classification. Positivity, neutrality, and negativity index, and synonym usage preference are good examples of semantic features. Distributed representation of words, Word2Vec, is also an attempt to extract and represent the semantic features of a word, sentence, and paragraph [8]. The usage of Word2Vec in authorship attribution tasks has not yet been studied explicitly. Due to the application domain dependency of Word2Vec features their usage will be introduced when discussing application specific feature sets.

3.4.1 Positivity and Negativity index

In order to understand the general mood and the preference of positive and negative sentence structure in each author's work, a positivity and negativity score has been calculated for the authors *David Lawder* and *Alexander Smith* for the documents in the RCV1 corpora. The code is shown in 3.5, whereas in 3.6 we can see the results that points out Alexander Smith writing more negative articles than David Lawder. Both of the authors wrote the majority of the articles classified as positive than negative.

Code Listing 3.5: Compute sentence Positivity and Negativity scores

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
# Pre-Processing
SIA = SentimentIntensityAnalyzer()
# Applying Model, Variable Creation
sentiment = pd.concat([df_david_lawder, df_alexander_smith])
sentiment['polarity_score']=sentiment.articles.apply(lambda x:SIA.
                                    polarity_scores(x)['compound'])
sentiment['neutral_score']=sentiment.articles.apply(lambda x:SIA.
                                    polarity_scores(x)['neu'])
sentiment['negative_score']=sentiment.articles.apply(lambda x:SIA.
                                    polarity_scores(x)['neg'])
```

```
sentiment['positive_score']=sentiment.articles.apply(lambda x:SIA.
                                    polarity_scores(x)['pos'])
sentiment['sentiment']=''
sentiment.loc[sentiment.polarity_score>0,'sentiment']='POSITIVE'
sentiment.loc[sentiment.polarity_score==0,'sentiment']='NEUTRAL'
sentiment.loc[sentiment.polarity_score<0,'sentiment']='NEGATIVE'
# Normalize for Size
auth_sent= sentiment.groupby(['author','sentiment'])[['articles']].
                                    count().reset_index()
for x in ['David Lawder', 'Alexander Smith']:
auth_sent.articles[auth_sent.author == x] = (auth_sent.articles[
                                    auth_sent.author == x]/\
auth_sent[auth_sent.author ==x].articles.sum())*100
ax= sns.barplot(x='sentiment', y='articles',hue='author',data=
                                    auth_sent)
ax.set(xlabel='Author', ylabel='Sentiment Percentage')
ax.figure.suptitle("Author by Sentiment", fontsize = 24)
plt.show()
```

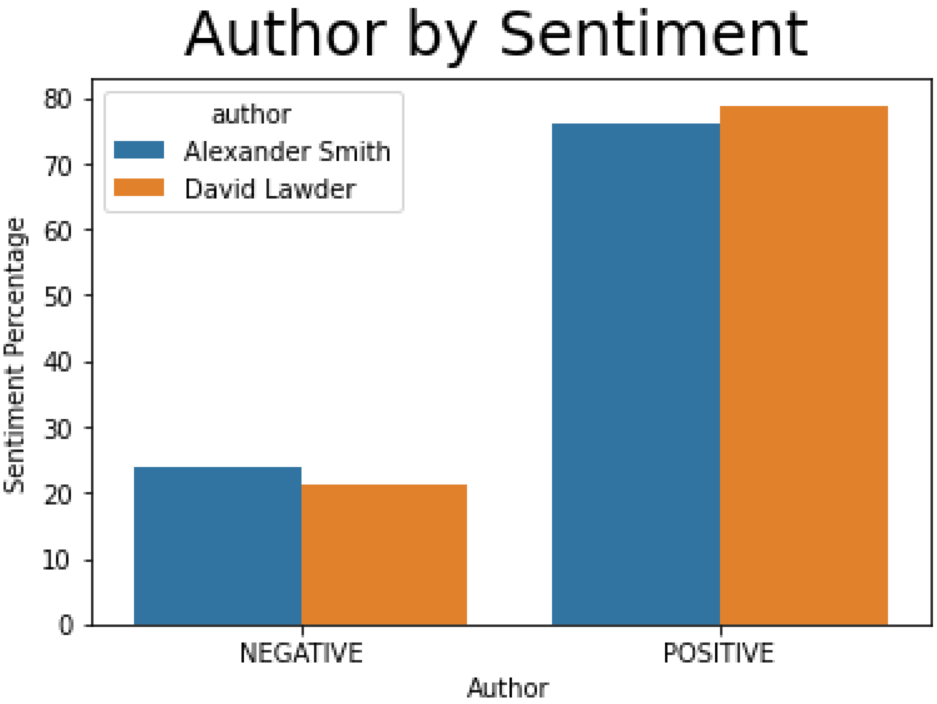


Figure 3.6: Bar Plot of sentiment analysis for 2 of the authors in the RCV1 CCAT_10 corpora across all their documents

3.5 Application Specific Features

When the application domain of the authorship attribution problems are different such as email messages or online forum messages, author style can be better characterized using structural, content specific, and language specific features. In such domains, the use of greetings and farewells, types of signatures, use of indentation, paragraph lengths, font color, font size could be good features [13].

3.5.1 Vector embeddings of words (Word2Vec)

Word2Vec[2] leverages the context of the target words. Essentially, we want to use the surrounding words to represent the target words with a Neural Network whose hidden layer encodes the word representation. Similar words are close to each other in the vector space. For example, it was shown in [10] that $vector[King] - vector[Man] + vector[Woman]$ results in the vector that is closest to the representation of the $vector[Queen]$. Figure 7.4 shows this representation in a simple way. The ways to make use of Word2Vec in the dataset is various. For example, a Word2Vec model can either be built by considering every authors text data separately, or can be imported using previously trained word vectors on other large text corpus. It can, then, be plotted into two dimensional vector space by using dimensionality reduction techniques (TSNE, for example[3]). We can also make use of pre-trained word vectors of Glove to see the difference of usages in such words between an author and a pre- trained word vector. Moving with the idea of training Word2Vec per author, one can also do a cosine distance measure for the same word or same sentence.

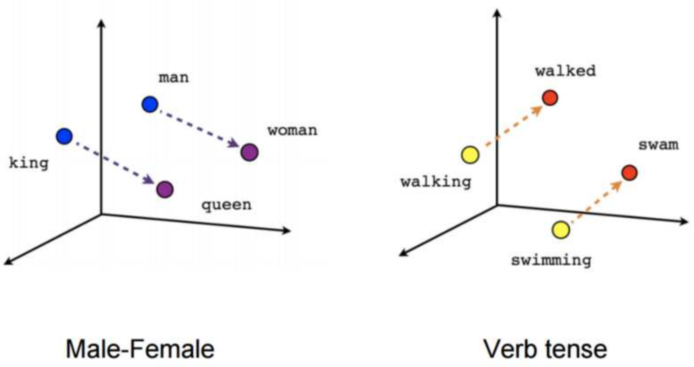


Figure 3.7: Examples of Word2Vec representation with vector distance

---

[2] `https://code.google.com/archive/p/word2vec/`

[3] t-Distributed Stochastic Neighbor Embedding (t-SNE) is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets

3.5.2 Vector embeddings of paragraphs(Sentence2Vec)

3.5.3 Vector embeddings of documents (Doc2Vec)

Distributed word representation in a vector space (word embeddings) is a novel technique that allows to represent words in terms of the elements in the neighborhood. Distributed representations can be extended to larger language structures like phrases, sentences, paragraphs and documents. The capability to encode semantic information of texts and the ability to handle high-dimensional datasets are the reasons why this representation is widely used in various natural language processing tasks such as text summarization, sentiment analysis and syntactic parsing.

Distributed representations can be extended to model not only words, but also larger language structures like phrases, sentences and documents [7].

CHAPTER 4

State of the art: Data collection and Techniques for Authorship Attribution

Word2vec (Mikolov et al., 2013a) has gained kinds of traction today. As the name shows, it translates words to vectors called word embeddings. That is to say, it gets the vector representations of words. We used gensim[1], a python tool, to get word2vec module, that uses by default Continuos Bag of Words (CBOW) method.

---

[1] `https://radimrehurek.com/gensim/`

4.1 SVM studies

4.1.1 SVM studies on authorship attribution

4.2 GDELT studies

4.2.1 Victorian era books

4.2.2 Authorship attribution GDELT

4.3 RCV1 studies

4.3.1 Studies on RCV1 on authorship attribution

4.4 The guardian studies

4.4.1 Cross-topic authorship attribution

4.5 Studies on Stanford Amazon Food Reviews

4.6 Dataset selection

The elements that have an impact on the performance of the model are the input corpora, model architecture and the hyper-parameters. In many works lemmatized, lowercased and shuffled input during training the word2vec are recommended; we carried out our experiments with these settings as detailed above.
For each corpus we used different approach for the documents vocabulary of the training model, such as:

- **ICD-9-CM Dictionary:** we used every entry of the dictionary as a single document
- **Wikipedia:** we used the entire content of each page as a single document
- **SDO:** we used every different diagnosis as a single document

We then made use of *gensim.utils.simple_preprocess*[2], which convert a document into a list of lowercase tokens, ignoring tokens that are too short or too long. In the end we trained each corpus separately forming 3 separate models and one joining all the documents of each corpus. Every model has been trained with the same hyperparameters:

---

[2] `https://radimrehurek.com/gensim/utils.html`

- min_count=2
  Ignores all words with total frequency lower than this.

- size=200
  Dimensionality of the word vectors.

- window=10
  Maximum distance between the current and predicted word within a sentence.

- workers=10
  Use these many worker threads to train the model (=faster training with multicore machines).

- epochs=10
  Number of iterations (epochs) over the corpus.

Code for training the models can be found in Appendix A.2.2.

If you parse emergency room discharge records data, you can easily notice how much noisy the data are. Especially dealing with such short sentence, it's very important to take action with misspelled words. Table 4.1 shows the first five diagnosis with at least a typo error beginning with the letter "a".

Table 4.1: First 5 diagnosis in the emergency room discharge records corpus that contains at least one typo error beginning with the letter "a".

trauma cranico non commotivo vasta flc del cuoio capelluto ferite escoriate multiple **aa** inferiori e superiori colpo di frusta del rachide cervicale
contusione aavampiede destro
frattura plurima ossa dorsali del naso con ferita lc piramide nasale contusione mano ginocchio dx **aabrasioni**
edema **aal** volto da probaile reazione allergica a ketoprofene oki
ferita profonda lacero contusa **aalla** base del dito mano ds

non-committal head injury extensive wound lacerated-bruised scalp multiple excoriate wounds **ls** upper and lower whiplash of the cervical spine
right **forefooot** bruise
multiple fracture of the dorsal bones of the nose with wound tear bruised pyramid nasal bruise hand right knee **aabrasions**
edema **iin** the face from probable allergic reaction to anti-inflammatory medicine
wound deep lacerated and bruised **aat** the base of the right hand finger

Word embeddings is generally a more powerful tool than to do auto-correction of misspelled words, but with our models we can do this as well. For example in Table 4.2, we can look for the most similar words to this italian misspelled words that we found in

diagnosis: *emoragia* (typo of bleeding) and *ati* (typo of limbs).
The reader has to understand that our model is unsupervised, when we will show closest word vectors to the given word it will be a result solely based on the context taken from the training dataset. In bold we point out the correct spelled corresponding word in italian.

Table 4.2: Most-similar words to: *emoragia* (typo of bleeding) and *ati* (typo of limbs).

emoragia	Value	ati	Value
emorraggia	0.762	**arti**	0.604
emorragia	0.751	tvparti	0.572
egdsemorragia	0.713	rti	0.564
emorrazgia	0.700	artitrauma	0.561
cureemorragia	0.634	artie	0.555

We can also point out that correct-spelled words are less common than misspelled words for the example taken in consideration. In Using PCA[3] we can see in Figure 4.1 the 2D representation of the top 10 most similar word vectors to the correct spelled italian word for diabetes. [See A.2.3 for code].
The first 10 most similar word vectors to *diabete* (diabetes) are indeed misspelled words of itself.

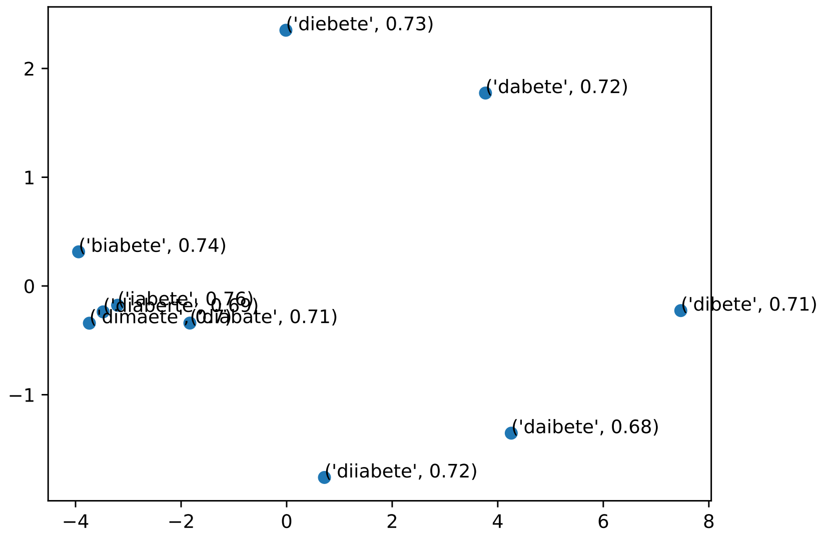


Figure 4.1: Top 10 most similar words to diabetes in the JoinData model.

Medical terminology is made up of numerous Greek and/or Latin suffixes and prefixes. It describes body parts, functions, surgical procedures, and is used in medical reports.

---

[3] Principal component analysis - is a dimension-reduction tool that can be used to reduce a large set of variables to a small set that still contains most of the information in the large set.

Every language has his own lingo and even every specific group of doctors has its own. When writing down an emergency room discharge records, doctors very often use abbreviations and sort of codes known specifically in the medical jargon to describe body parts and common words. This is done for plenty of reasons: first of all in Italy doctors still use quite a lot handwritten emergency room discharge records, but even typing on a keyboard doctors are known to be a little bit lazy. From our evaluated models we can look for meaning of the main common medical abbreviation and on the other hand knowing how a word is commonly abbreviated by doctors.
For example we can look for the most similar words to this italian words: *tac* (CT scan), *radiografia* (radiography), *paziente* (patient).

Table 4.3: Looking for medical jargon of this words: *tac* (CT scan), *radiografia* (radiography), *paziente* (patient)

tac	**Value**	**radiografia**	**Value**	**paziente**	**Value**
tc	0.869	**rx**	0.604	**pz**	0.856
rmn	0.675	lastra	0.572	paz	0.695
angiotc	0.631	computerizzata	0.564	soggetto	0.689
rnm	0.567	radiologico	0.561	bambino	0.451
ecografia	0.556	tac	0.555	persona	0.405

In table 4.3 we showed most similar words tested with the domain-specific model joining all the three medical specific corpora data. We can see that in medical jargon '*tac*' (CT scan) you are more likely to find it in Emergency Room discharge records as '*tc*'. The same for '*radiografia*' (radiography) and '*paziente*' (patient), that doctors use to abbreviate respectively '*rx*' and '*pz*'.

We can also revert the process and look for the meaning of this abbreviations and words in medical jargon: '*als*' (acronym of Advanced Life Support) and '*pz*' (abbreviation of patient).

Table 4.4: Looking for the meaning of this words in medical jargon: '*als*' (acronym of Advanced Life Support) and '*pz*' (abbreviation of patient)

als	**Value**	**pz**	**Value**
defibrillazione	0.614	**paziente**	0.856
acls	0.584	paz	0.856
rianimatorie	0.583	soggetto	0.519
asistolia	0.567	schziofrenia	0.442
rianimazione	0.525	pazinete	0.374

From table 4.4 we can learn some useful information; first of all we can see that '*pz*' (abbreviation of patient) corresponds indeed to the word '*paziente*' (patient). Moreover, looking for the acronym '*als*' solely based on the context we can guess that it's something about defibrillator and life's saving; as a matter of fact the most similar words to '*als*' are '*acls*' (Advanced Course Life Support) and '*defibrillazione*' (Defibrillation).

We can see in chart 4.2 the 2D vectors representation of the 10 most similar words to '*paziente*' (patient)

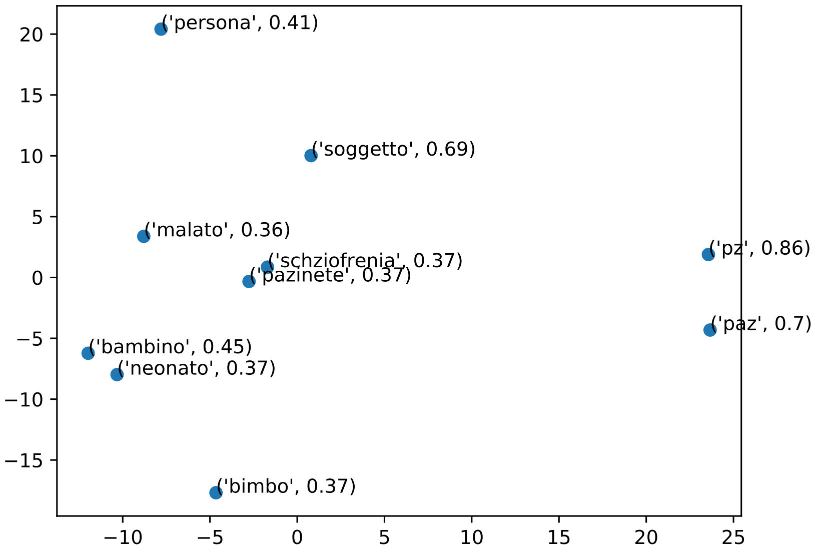


Figure 4.2: Top 10 most similar words to *paziente* (patient) in the JoinData model.

Figure 4.3 shows top ten most frequent words in the emergency room discharge records corpus. As we can see the words are: *cardiopatia* (hearth disease), *dolore* (pain), *addominale* (abdomen), *ipertensione* (hypertension), *dx* (abbreviation of right), *pz* (abbreviation of patient), *diabete* (diabetes), *paziente* (patient), *trauma* (trauma) and *cronica* (chronic). The word *cardiopatia* shows up **92'639** times (on **20'281'315** total words) in the diagnosis.

According to data obtained under the 'CUORE' project[4], applying the incidence estimates on the population aged 35-74 years recorded by the Istat[5] 2001 Census, the number of new coronary events is around 80,000 per year in men and 20,000 per year for women. In addition, the hearth diseases are placed, according to a study of Istat[5] on the first 25 causes of death in 2013-2014, among the top 3 causes of death in Italy.

Figure 4.4 shows top ten most frequent words in the Wikipedia health-related corpus. As we can see the words are: *essere* (to be), *altri* (others), *viene* (comes), *portale* (portal),

---

[4] `http://www.salute.gov.it/imgs/C_17_navigazioneSecondariaRelazione_1_listaCapitoli_capitoliItemName_1_scarica.pdf`

[5] Italian Statistic National Institute

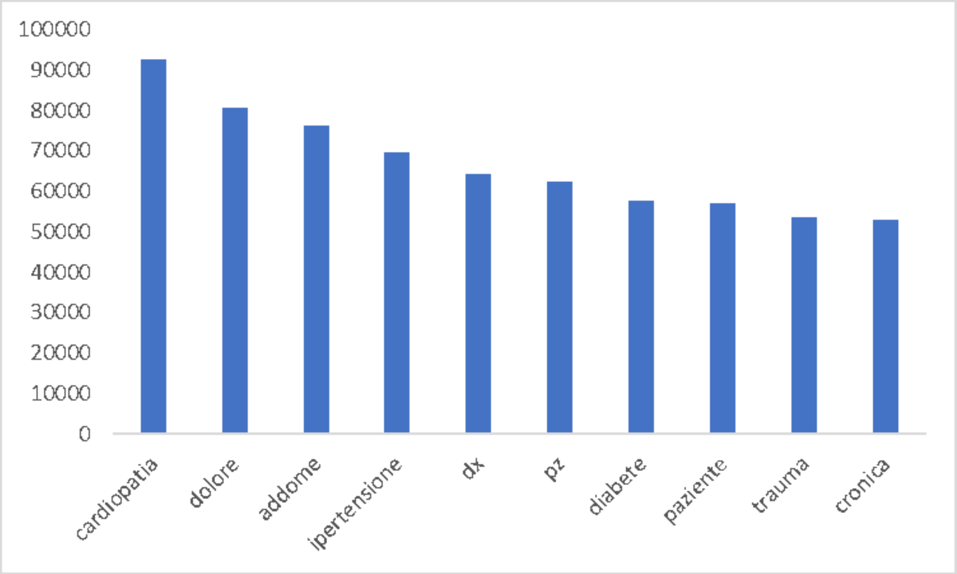


Figure 4.3: Top 10 most frequent words in the emergency room discharge records corpus.

consultato (consulted), *solo* (only), *possono* (can), *medicina* (medicine), *parte* (part) and isbn[6]. The word *essere* (to be) shows up **36484** times (on **13'195'758** total words) in the corpus data.

This is a very common word in Wikipedia, placed among the top 62 in all Wikipedia rank.[7] Indeed the word *essere* (to be) shows 2'042 millions of times in all italian Wikipedia pages. One thing that is unpredictable in Wikipedia is that is not an official source of data, so basically it is up to who edits a Wikipedia page to write in his/her own style.

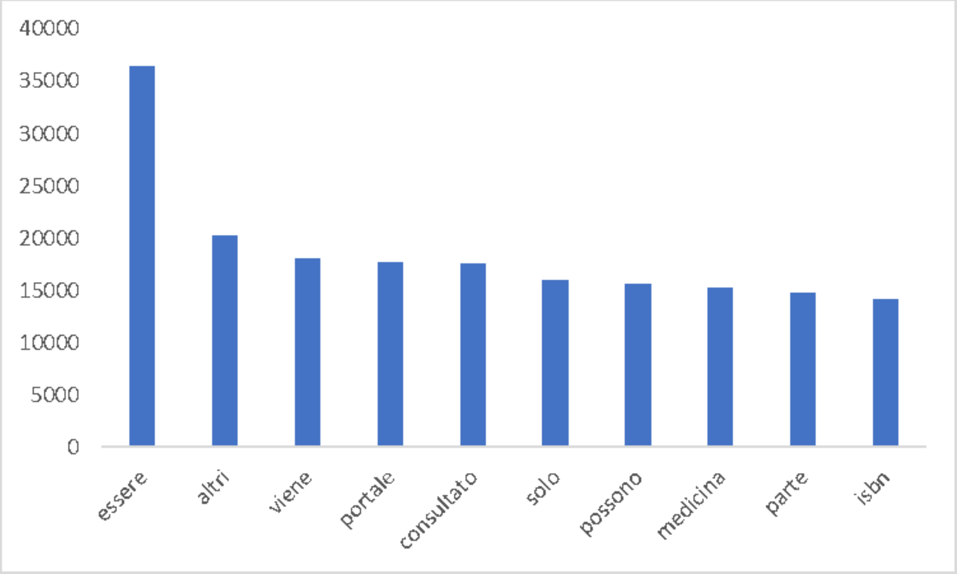


Figure 4.4: Top 10 most frequent words in the Wikipedia health-related corpus.

---

[6] International Standard Book Number

[7] `https://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/Italian1000`

Figure 4.5 shows top ten most frequent words in the dictionary. As we can see the words are: *altre* (other), *specificata* (specified), *senza* (without), *menzione* (mention), *frattura* (fracture), *perdita* (loss), *tumori* (tumors), *traumatismo* (traumas), *coscienza* (consciousness) and *condizione* (condition). The word *altre* (other) shows up **1761** times (on **130'218** total words) in the vocabulary.
This is intuitively obvious, as ICD-9-CM contains more than 16 thousand classes while ICD-10-CM contains more than 60 thousands. Since even ICD-10-CM is not exhaustive, so many diseases will be specified under the category code that cites the word *altre* (others) in the definition. Of course this still applies to ICD-9-CM that contains 1 on six specific definitions compared to ICD-10-CM.
Table 4.5 shows first 5 coded diagnosis that contains the word *altre* (others) in the definition's vocabulary.

Table 4.5: First 5 ICD-9-CM coded diagnosis that contains the word *altre* (other) in the definition's vocabulary.

003 - **Altre** infezioni da Salmonella
00329 - **Altre** infezioni localizzate da Salmonella
0038 - **Altre** infezioni specifiche da Salmonella
0048 - **Altre** infezioni specifiche da Shigella
005 - **Altre** intossicazioni alimentari (batteriche)

003 - **Other** salmonella infections
00329 - Local salmonella inf NEC
0038 - Salmonella infection NEC
0048 - Shigella infection NEC
005 - **Other** food poisoning (bacterial)

We trained 4 models one for each corpus data and one joining all health-specific datasets calling it 'JoinData' model. Each corpus is essential to result in a complete health-specific word embeddings.

This corpus data is essential for plenty of reasons. First of all, we have a lot of examples of how doctors write diagnosis and emergency room discharge records. Secondly we have all the medical jargon (or lingo) that in Italy, and especially in Forlì Hospital, doctors use. Moreover, we can take in consideration all the typo errors and common mistakes doctors often do when writing discharge records. Everything that is in this corpus is essential for the real approach to the diagnosis.

This corpus is the most used corpus data for word embeddings. It's easy to download, there's no copyright issues and also it's available in almost every language you need. Downloading only the health-specific pages increased not only the quantity of words vectors in the word embedding but also the correct spelled medical words and some other non-doctors spoken terms that can be useful especially to those who aren't medical

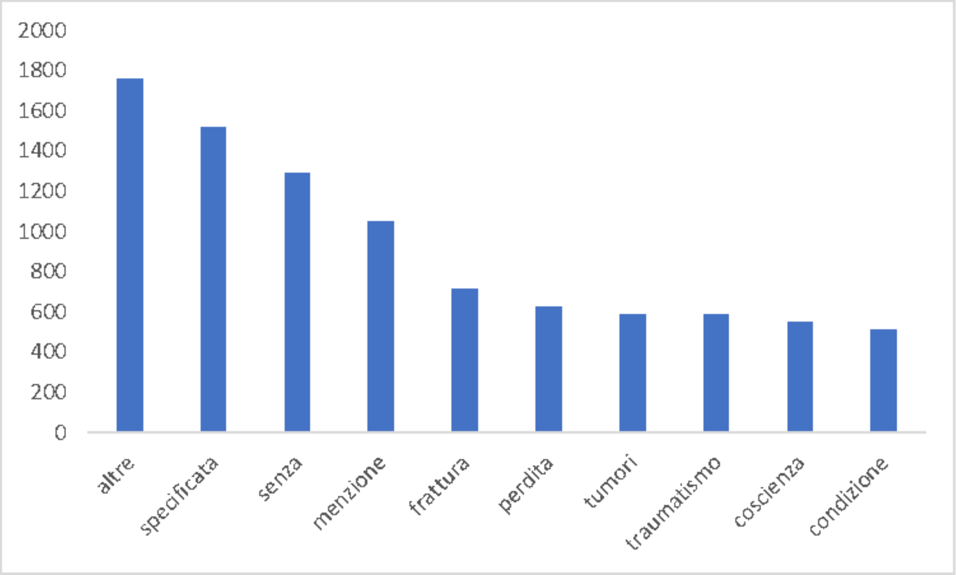


Figure 4.5: Top 10 most frequent words in the dictionary corpus.

expert.

This corpus is essential for 2 main reasons: first it let us have the right terms and definition of each diagnosis and secondly we can have in the embeddings all the possible diagnosis, especially those who aren't likely to be found maybe because of Forlì Hospital isn't that big or because in Italy we haven't some diseases due to their rareness.

Our main purpose is to show that domain-specific word embeddings is actually better than general word embedding trained with a big data corpus.
As a matter of fact what we can show is trying to find out most similar words to health-related terms and see which words are more close to the meaning in each model. We expect that our 'JoinData' model that was trained with emergency room discharge records, wikipedia health-specific pages and the ICD-9-CM dictionary performs better than any other separate model and all of them performs better than a general model.
We choose to evaluate the word: *arteria* (artery) and *frattura* (fracture).
As we can see from table 4.6 most similar words are awful results. Most certainly, the TED corpus data used to train the model didn't have medical terms or data.

Table 4.7 shows that wikipedia data are actually better than TED corpus. Wikipedia is often used for training word embeddings models and our corpus built with petscan and querying the wikimedia API is of course a subset of the general one. So we expect that table 4.7 and table 4.10 perform with similar results. We can see this guess is true by the plotting of the word vectors of the top 10 most similar word to *arteria* (artery) in figure 4.7 and 4.10.

By comparing tables 4.6 and 4.7 with our best domain-specific model that shows its results in table 4.8 we can confirm that domain-specific models are actually more precise

Table 4.6: Most-similar words to: *arteria* (artery) and *frattura* (frattura) in the TED corpus model.

Arteria	**Value**	**Frattura**	**Value**
l'omosessualità	0.572	un'arteria	0.510
davanzale	0.538	difetto	0.490
indirizzano	0.532	pretesto	0.479
l'interferenza	0.530	turchese	0.467
immutato	0.526	mina	0.464

Table 4.7: Most-similar words to: *arteria* (artery) and *frattura* (frattura) in the general Wikipedia corpus model.

Arteria	**Value**	**Frattura**	**Value**
arterie	0.814	clavicola	0.703
succlavia	0.712	fratture	0.700
mesenterica	0.706	perone	0.664
gastroepiploica	0.705	lussazione	0.663
l'arteria	0.700	rottura	0.662

than the general one.

Table 4.8: Most-similar words to: *arteria* (artery) and *frattura* (frattura) in the JoinData model.

Arteria	**Value**	**Frattura**	**Value**
vena	0.578	infrazione	0.756
artria	0.537	frattuta	0.578
ostio	0.506	dolentefrattura	0.571
embolizzazione	0.484	lussazion	0.555
auricola	0.481	fratture	0.544

Having a look at tables 4.11 and 4.9 we can see how specific corpus word embeddings performs. As a matter of facts, we see that for the word embedding trained only with ICD-9-CM dictionary data, most-similar words are all medical terms that you can find in the official diagnosis provided by the italian Minister of Health. The first result *carotide* (carotid artery) is actually the most important artery in our body.
For what concern the model trained with only emergency room discharge records, we expect unofficial medical terms, written in medical jargon or with typo errors. Indeed, the top result for artery is: *artria* (typo error of artery), an evident typo of *arteria* (artery). We can look at the top 10 most similar words vectors to *artery* (artery) in Figure 4.11.

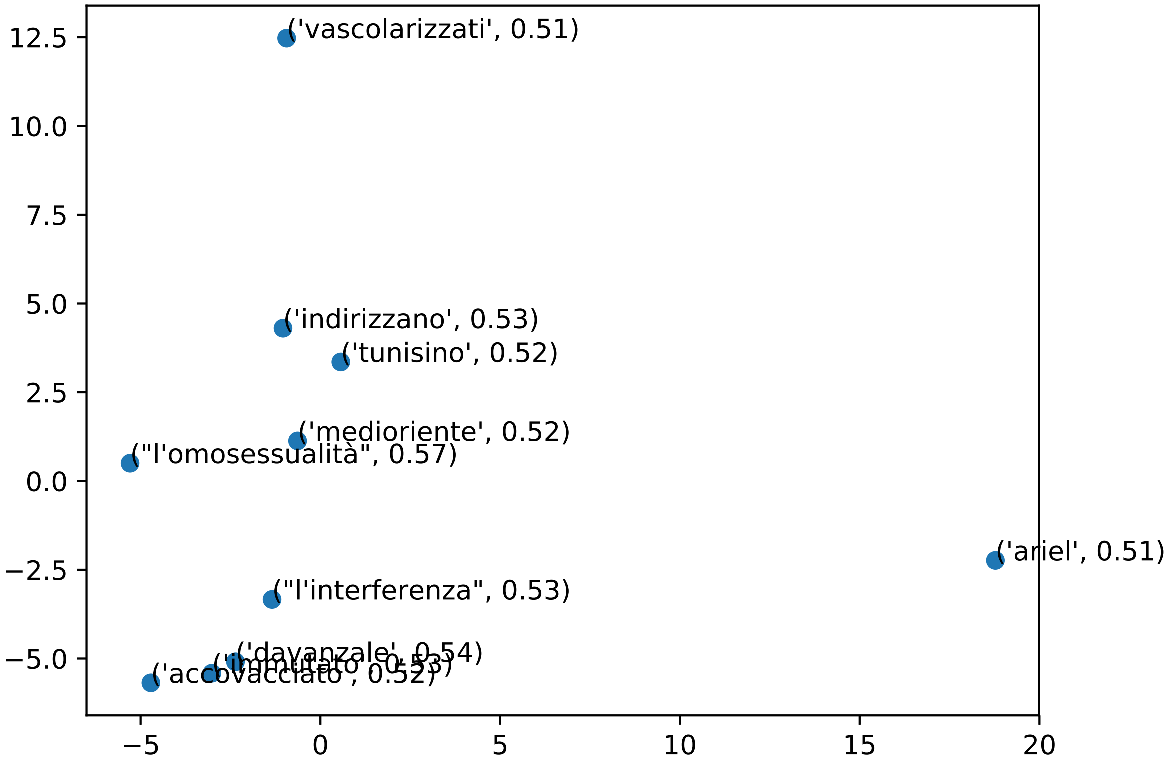


Figure 4.6: Plotting 10 most similar word vectors to *arteria* (artery) in the Ted Corpus model.

Table 4.9: Most-similar words to: *arteria* (artery) and *frattura* (frattura) in the dictionary data model.

Arteria	**Value**	**Frattura**	**Value**
carotide	0.935	base	0.884
vena	0.875	chiusa	0.877
basilare	0.871	volta	0.871
occlusione	0.869	cranica	0.866
aneurisma	0.856	diafisi	0.856

Table 4.10: Most-similar words to: *arteria* (artery) and *frattura* (frattura) in the Wikipedia health-specific corpus model.

Arteria	**Value**	**Frattura**	**Value**
aorta	0.662	tibia	0.664
vena	0.629	lussazione	0.662
biforcazione	0.623	deformazione	0.632
aneurisma	0.588	deformità	0.597
uretere	0.587	fratture	0.588

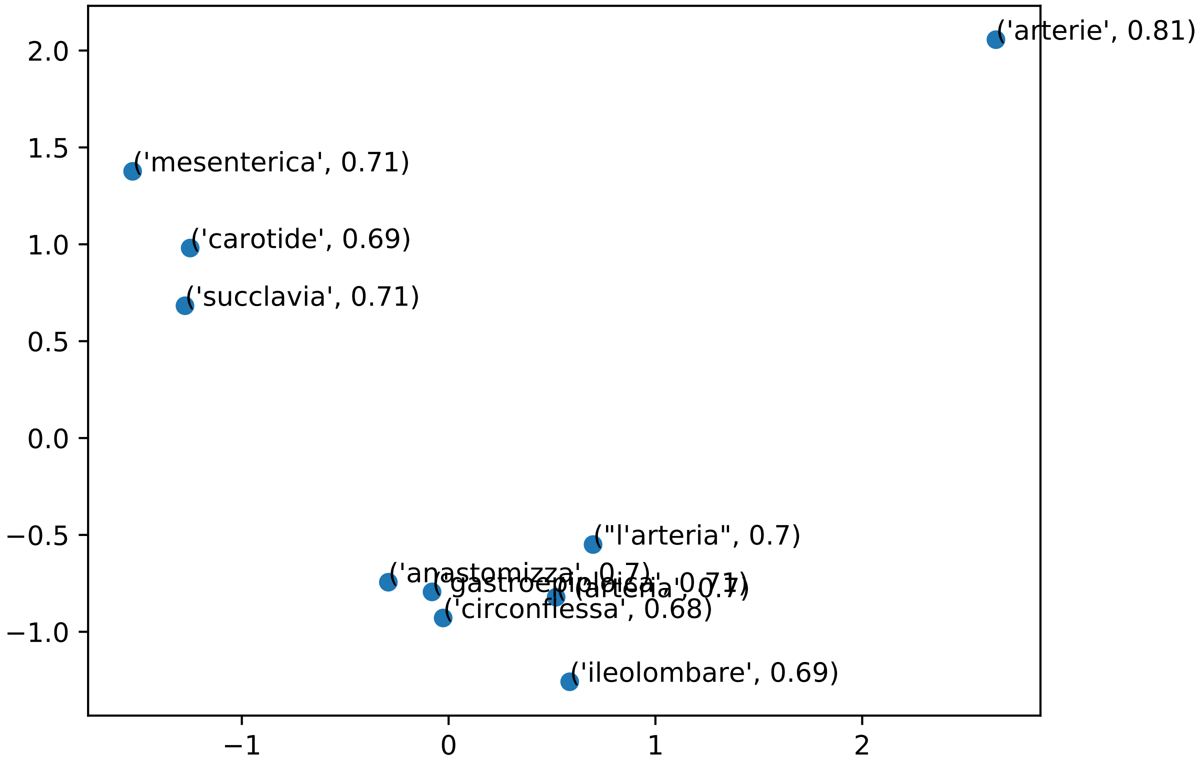


Figure 4.7: Plotting 10 most similar word vectors to *arteria* (artery) in the general Wikipedia corpus model.

Table 4.11: Most-similar words to: *arteria* (artery) and *frattura* (frattura) in the emergency room discharge records corpus model.

Arteria	Value	Frattura	Value
artria	0.550	infrazione	0.756
ostio	0.506	fratture	0.493
auricola	0.496	frattuta	0.477
comune	0.473	aafrattura	0.458
pta	0.472	fratura	0.452

In a related work, we tried to automatically assign the codes corresponding to a textual diagnosis through a classifier. This classifier used our domain-specific word embedding for weighting words instead of using a general-purpose one and noticed a real gain in terms of accuracy.

This supervised classifier was trained using a 14-thousands-diagnosis dataset (that is just a tiny subset of the more than 700 thousands diagnosis collected from Forlì emergency room) manually labeled by qualified personnel.

Textual diagnoses were first preprocessed in order to minimize spelling errors and meaningless symbols. In this process, we built a token dictionary of words contained in the

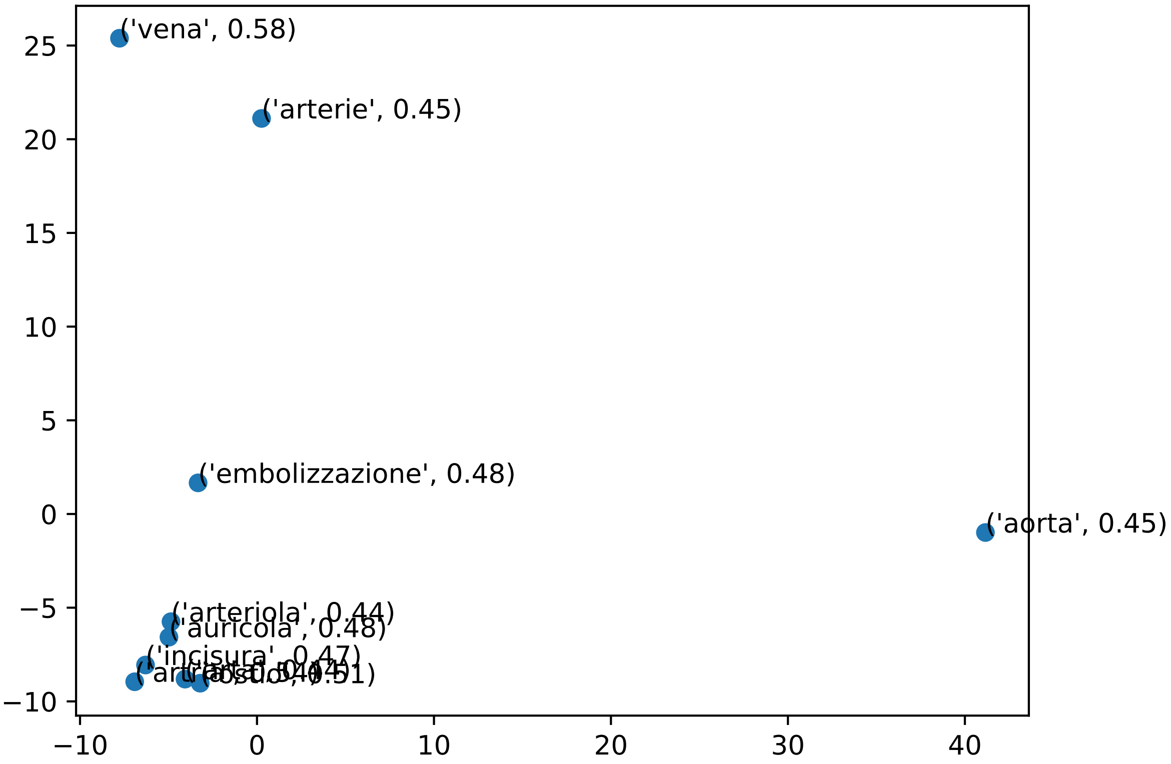


Figure 4.8: Plotting 10 most similar word vectors to *arteria* (artery) in the JoinData model.

training set.

Subsequently we created an embedding matrix where multidimensional vector were associated to each word of each diagnosis. To build the embedding matrix we start from a matrix of dimension = (number of words in the diagnosis, length of the embedding vector) where each element is initialized to 0. The matrix is progressively filled by iterating on the tokenizer indexes, where each row corresponds to an index(and therefore to a corresponding word) and its content is the word vector embedded. This embedding matrix is used to associate a certain weight with words of the input diagnosis to the classifier in order to capture the semantics of sentences and increase final precision.

We computed the accuracy of each word embedding through the f1 score[8] of the built classifier. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal.

The formula for the F1 score is:

```
F1 = 2 * (precision * recall) / (precision + recall)
```

where:

---

[8] `http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html`

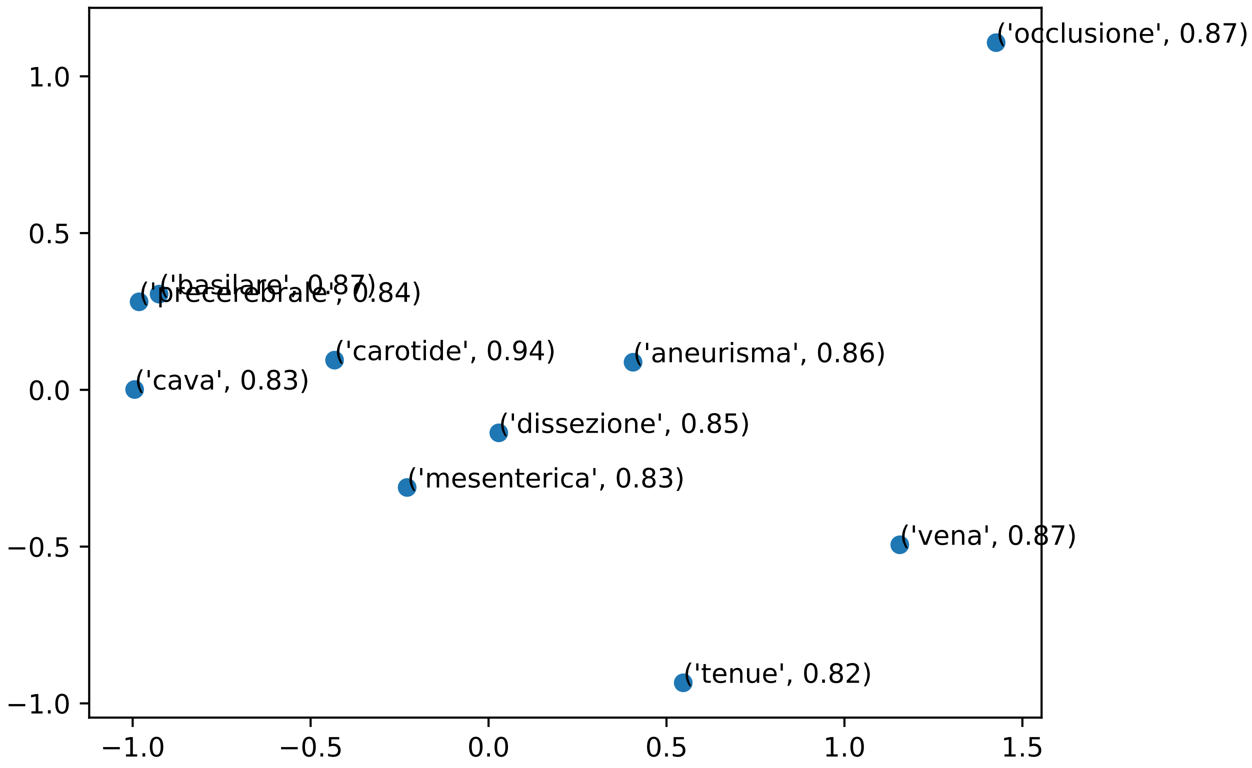


Figure 4.9: Plotting 10 most similar word vectors to *arteria* (artery) in the Dictionary data model.

- **Precision:** is the fraction of relevant instances among the retrieved instances.
- **Recall:** is the fraction of relevant instances that have been retrieved over the total amount of relevant instances.

We can see in table 4.12 results of F1 micro[9] and table 4.13 shows the results of F1 weighted[10].

Table 4.12: F1 Micro of each word embedding, both domain-specific and general purpose

Word embeddings	**Value**
JoinData	**0.18**
Emergency room discharge records	0.170
Wikipedia health-specific	0.110
Wikipedia general	0.050
Dictionary ICD-9-CM	0.040
TED corpus	0.040

---

[9] Calculate metrics globally by counting the total true positives, false negatives and false positives.

[10] Calculate metrics for each label, and find their average weighted by support (the number of true instances for each label). This alters ‘macro’ to account for label imbalance; it can result in an F-score that is not between precision and recall.

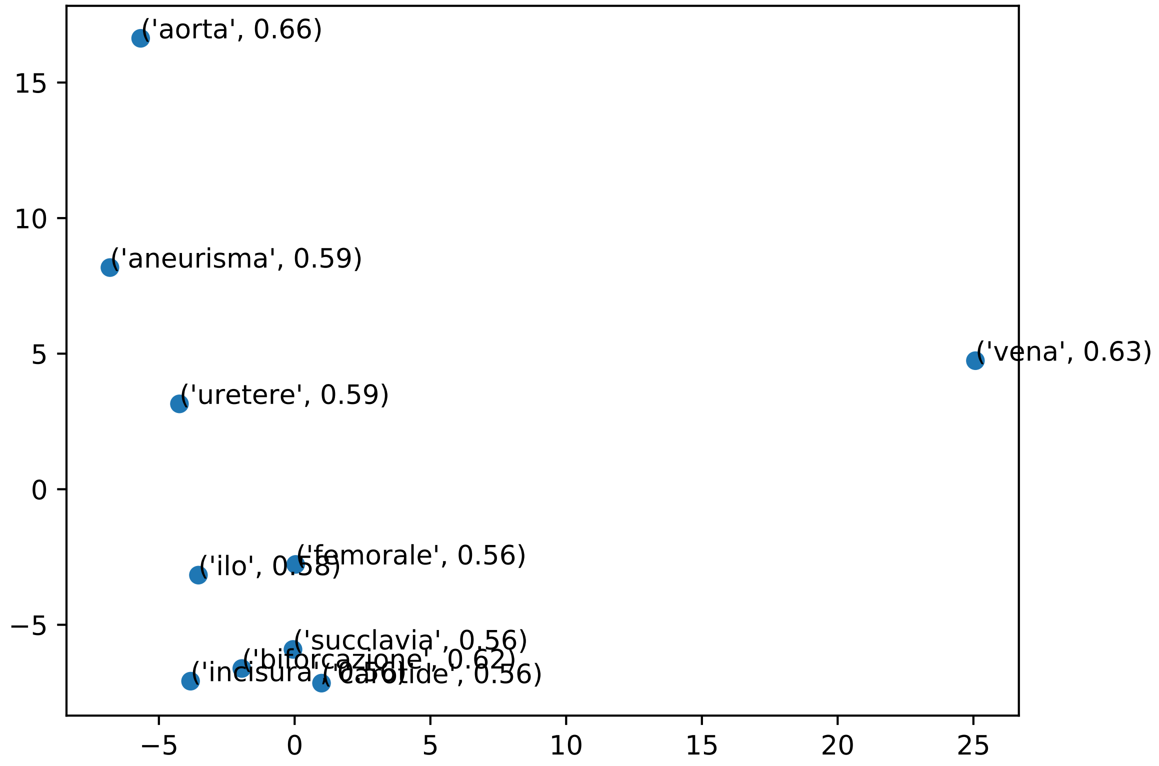


Figure 4.10: Plotting 10 most similar word vectors to *arteria* (artery) in the Wikipedia health-specific data model.

Table 4.13: F1 Weighted of each word embedding, both domain-specific and general purpose

Word embeddings	**Value**
JoinData	**0.233**
Emergency room discharge records	0.210
Wikipedia health-specific	0.175
Wikipedia general	0.129
TED corpus	0.106
Dictionary ICD-9-CM	0.105

As we can clearly see from this results, we can conclude that domain-specific word embeddings is more suitable for this task. Moreover the JoinData model is undoubtedly the most accurate among the other word embeddings, mostly over the general purpose word embeddings.

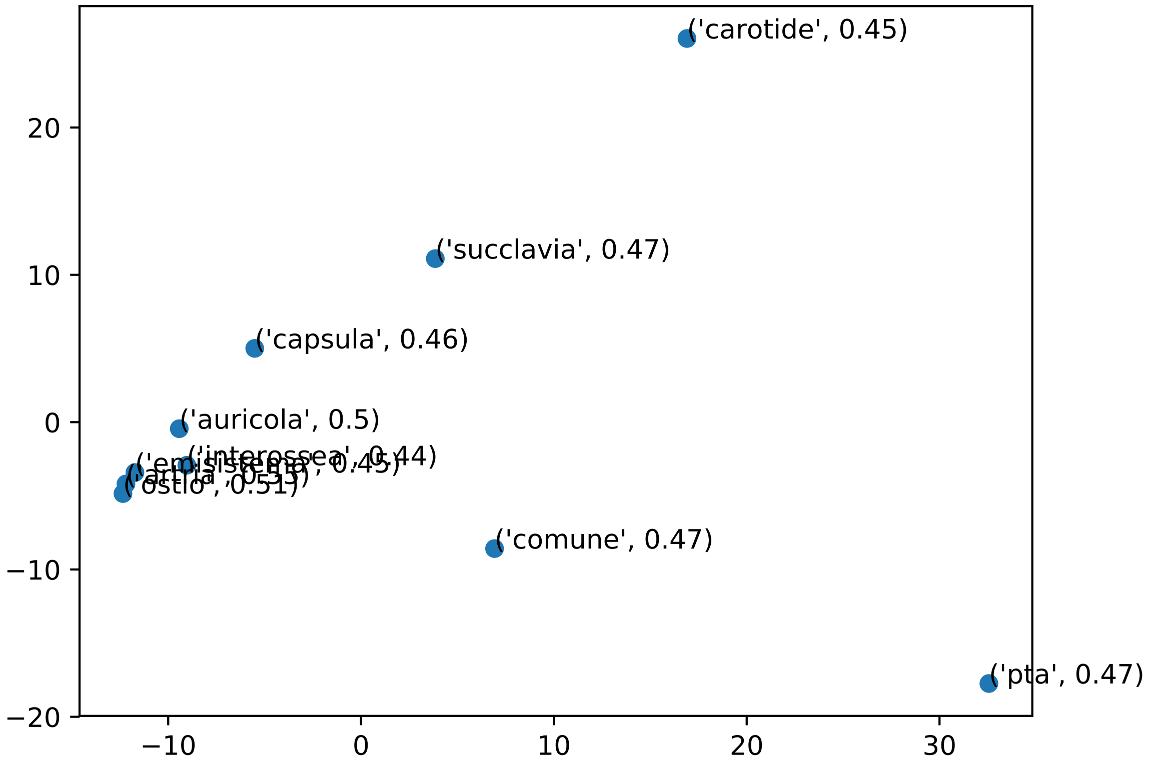


Figure 4.11: Plotting 10 most similar word vectors to *arteria* (artery) in the emergency room discharge records corpus model.

CHAPTER 5

OUR APPROACH

CHAPTER 6

RESULT AND EVALUATION

In this thesis we proposed domain-specific word embeddings for ICD-9-CM classification. In particular, we showed how to collect the dataset and enlarge it with other medical data.
We gathered together 3 main corpora of specific medical data: the first one from the emergency room discharge records of the Forlì Hospital, where we collected more than 700k real anonymous diagnosis written by doctors when sending home patients, the second one has been downloaded from all the italian medical articles available on Wikipedia and the last one was the official ICD-9-CM dictionary of the more than 16k definitions of diagnosis and their corresponding code.
For the general and comparison models, as we weren't able to use news data (i.e. GoogleNews, the most popular) because they aren't provided in italian language, we were forced to use other pre-trained models in italian. We found 2 of them with different backgrounds: the first one is the wikipedia dump of all the pages in it, the other one is provided with TED speech corpus translated in italian.
Due to the completely different nature of the two general purpose datasets, they showed when testing most similar words that they responded in a completely different way because they aren't filled with medical specific words in their starting dataset.
Subsequently, with the help of gensim tool and Word2Vec we trained each corpus separately forming 3 separate models and one joining all the documents of each corpus. Every model has been trained with the same hyperparameters. We showed the results of the training of this dataset compared to the general purpose retrieved from TED corpus and general Wikipedia italian dump. Through most-similar words tool with the help of 2D plot, we tested the different models and showed how they respond.
We pointed out some of the most common feature typical in a medical text, such as typo errors and medical jargon. Medical terminology is made up of numerous Greek and/or Latin suffixes and prefixes. It describes body parts, functions, surgical procedures, and

is used in medical reports. Every language has his own lingo and even every specific group of doctors has its own. When writing down an emergency room discharge records, doctors very often use abbreviations and sort of codes known specifically in the medical jargon to describe body parts and common words. This is done for plenty of reasons: first of all in Italy doctors still use quite a lot handwritten emergency room discharge records, but even typing on a keyboard doctors are known to be a little bit lazy.

From our evaluated models we looked for the meaning of main commons medical abbreviations and on the other hand we discovered how a word is commonly abbreviated by doctors.

The reader has to understand that this process is automatic and the model was unsupervised. Most similar words to a given word it is a result solely based on the context taken from the training dataset.

We finally proofed that our domain-specific word embeddings should be chosen over the general purpose. We presented a related work where we tried to automatically assign the codes corresponding to a textual diagnosis through a classifier. This classifier used our domain-specific word embedding for weighting words instead of using a general-purpose one and we noticed a real gain in terms of accuracy.

As we can clearly see from this last results, we can conclude that domain-specific word embeddings is more suitable for this task. Moreover the ‘JoinData’ model is undoubtedly the most accurate among the other word embeddings, mostly over the general purpose word embeddings.

As for what concerns the classification problem of ICD-9-CM it’s still an hard task to solve, but we can unmistakably state that for word embedding of this type in a domain-specific context, such as the medical one filled with jargon, technique words and mostly typos, a domain-specific word embeddings performs better than a general purpose one.

CHAPTER 7

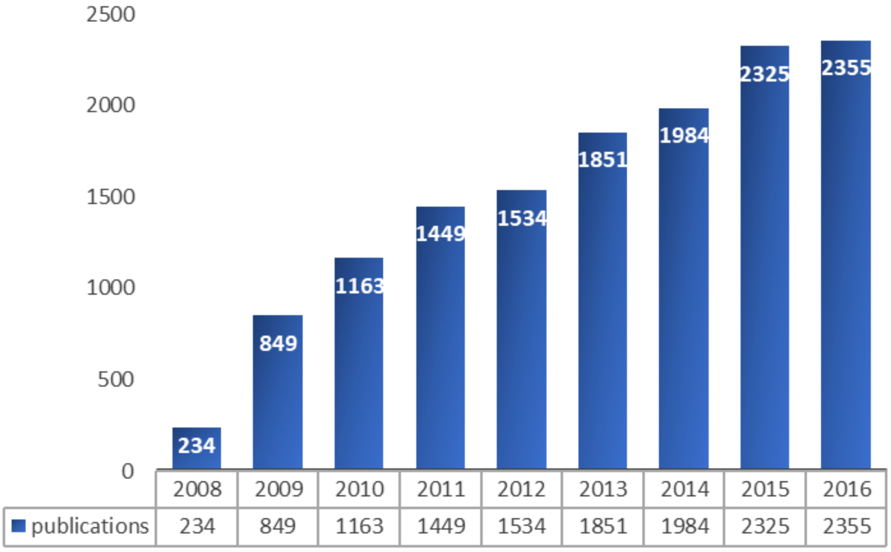
FUTURE WORKS

The baseline of the text mining is to obtain little pieces of desired information over tons of text data without having to read everything.

The vast numbers of biomedical text provide a rich source of knowledge for medical research. Text mining can help us to mine information and knowledge from a mountain of text and it is now widely applied. As shown in Figure 7.1, the number of publications obtained from PubMed using '*text mining*' as the query word in the title or abstract has grown substantially since 2000. Many researchers have taken advantage of text mining technology to discover novel knowledge to improve the development of biomedical research. [?]

The purpose of Text Mining is to process unstructured (textual) information, extract meaningful numeric indices from the text, and, thus, make the information contained in the text accessible to the various data mining (statistical and machine learning) algorithms. Information can be extracted to derive summaries for the words contained in the documents or to compute summaries for the documents based on the words contained in them. Hence, you can analyze words, clusters of words used in documents, etc., or you could analyze documents and determine similarities between them or how they are related to other variables of interest in the data mining project. In the most general terms, text mining will 'turn text into numbers' (meaningful indices), which can then be incorporated in other analyses such as predictive data mining projects, the application of unsupervised learning methods (clustering), etc. These methods are described and discussed in great detail in the comprehensive overview work by **?** **?** .

Text mining employs many computational technologies, such as machine learning,



	2008	2009	2010	2011	2012	2013	2014	2015	2016
■ publications	234	849	1163	1449	1534	1851	1984	2325	2355

Figure 7.1: The number of publications in PubMed using the query word "*text mining*" or "*literature mining*" in the title or abstract over the last years. Search detail: *text mining* [Title/Abstract] or *literature mining* [Title/Abstract].

natural language processing, biostatistics, information technology, and pattern recognition, to find new exciting outcomes hidden in unstructured biomedical text. The goal of text mining is to derive implicit knowledge that hides in unstructured text and present it in an explicit form. This generally has four phases: information retrieval, information extraction, knowledge discovery, and hypothesis generation. Information retrieval systems aim to get desired text on a certain topic; information extraction systems are used to extract predefined types of information such as relation extraction; knowledge discovery systems help us to extract novel knowledge from text; hypothesis generation systems infer unknown biomedical facts based on text. Thus, the general tasks of biomedical text mining include information retrieval, named entity recognition and relation extraction, knowledge discovery and hypothesis generation. [**?**]

To reiterate, text mining can be summarized as a process of 'numericizing' text. At the simplest level, all words found in the input documents will be indexed and counted in order to compute a table of documents and words, i.e., a matrix of frequencies that enumerates the number of times that each word occurs in each document. This basic process can be further refined to exclude certain common words such as 'the' and 'a' (stop word lists) and to combine different grammatical forms of the same words such as 'traveling', 'traveled', 'travel', etc. (This process is known as stemming[1]). However, once a table of (unique) words (terms) by documents has been derived, all standard statistical and data mining techniques can be applied to derive dimensions or clusters of words or documents, or to identify 'important' words or terms that best predict another outcome variable of interest.

---

[1] The term stemming refers to the reduction of words to their roots so that, for example, different grammatical forms or declinations of verbs are identified and indexed (counted) as the same word. For example, stemming will ensure that both 'travel' and 'traveled' will be recognized by the program as the same word.

Once the input documents have been indexed and the initial word frequencies (by document) computed, a number of additional transformations can be performed to summarize and aggregate the information that was extracted. As described above, the most basic result of the initial indexing of words found in the input documents is a frequency table with simple counts, i.e., the number of times that different words occur in each input document. Usually, we would transform those raw counts to indices that better reflect the (relative) 'importance' of words and/or their semantic specificity in the context of the set of input documents (see the discussion of inverse document frequencies, above). A common analytic tool for interpreting the 'meaning' or 'semantic space' described by the words that were extracted, and hence by the documents that were analyzed, is to create a mapping of the word and documents into a common space, computed from the word frequencies or transformed word frequencies (e.g., inverse document frequencies).

After significant (e.g., frequent) words have been extracted from a set of input documents, and/or after singular value decomposition has been applied to extract salient semantic dimensions, typically the next and most important step is to use the extracted information in a data mining project.

- **Graphics (visual data mining methods).** Depending on the purpose of the analyses, in some instances the extraction of semantic dimensions alone can be a useful outcome if it clarifies the underlying structure of what is contained in the input documents. For example, a study of new car owners' comments about their vehicles may uncover the salient dimensions in the minds of those drivers when they think about or consider their automobile (or how they 'feel' about it). For marketing research purposes, that in itself can be a useful and significant result. You can use the graphics (e.g., 2D scatterplots or 3D scatterplots) to help you visualize and identify the semantic space extracted from the input documents.

- **Clustering and factoring.** You can use cluster analysis methods to identify groups of documents (e.g., vehicle owners who described their new cars), to identify groups of similar input texts. This type of analysis also could be extremely useful in the context of market research studies, for example of new car owners. You can also use Factor Analysis and Principal Components and Classification Analysis (to factor analyze words or documents).

- **Predictive data mining.** Another possibility is to use the raw or transformed word counts as predictor variables in predictive data mining projects.

In Numeric Natural Language Processing (NLP), we often map words into vectors that contains numeric values so that machine can understand it. Word embedding is a type of mapping that allows words with similar meaning to have similar vectorial representation.

A traditional way of representing words is one-hot vector, which is essentially a vector with only one target element being 1 and the others being 0. The length of the vector is equal to the size of the total unique vocabulary in the corpora. Conventionally, these unique words are encoded in alphabetical order. Namely, you should expect the one-hot vectors for words starting with "a" with target "1" of lower index, while those for words beginning with "z" with target "1" of higher index.

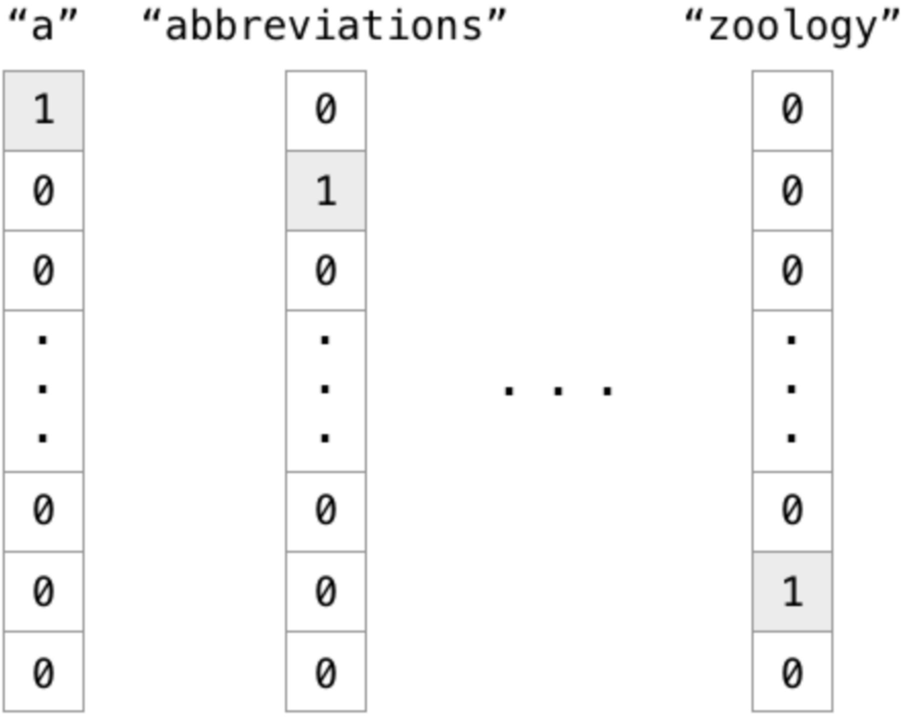


Figure 7.2: One Hot Vector: A simple and easy way to implement word vectors. 1 Bit set to 1 and all the others to 0. The dimension of the vector depends on the size of the vocabulary in input.

Though this representation of words is simple and easy to implement, there are several issues. First, you cannot infer any relationship between two words given their one-hot representation. For instance, the word "endure" and "tolerate", although have similar meaning, their targets "1" are far from each other. In addition, sparsity is another issue as there are numerous redundant "0" in the vectors. This means that we are wasting a lot of space.We need a better representation of words to solve these issues.

Word2Vec[2] is an efficient solution to these problems, which leverages the context of the target words. Essentially, we want to use the surrounding words to represent the target words with a Neural Network whose hidden layer encodes the word representation.

---

[2] `https://code.google.com/archive/p/word2vec/`

There are two types of Word2Vec, Skip-gram and Continuous Bag of Words (CBOW). I will briefly describe how these two methods work in the following paragraphs.

For skip-gram, the input is the target word, while the outputs are the words surrounding the target words. For instance, in the sentence "I have a cute dog", the input would be "a", whereas the output is "I", "have", "cute", and "dog", assuming the window size is 5. All the input and output data are of the same dimension and one-hot encoded. The network contains 1 hidden layer whose dimension is equal to the embedding size, which is smaller than the input/output vector size. At the end of the output layer, a softmax activation function is applied so that each element of the output vector describes how likely a specific word will appear in the context. In mathematics, the softmax function, or normalized exponential function is a generalization of the logistic function that *squashes* a K-dimensional vector z of arbitrary real values to a K-dimensional vector δ(z) of real values, where each entry is in the range (0,1) and all the entries add up to 1. The target is a (K-1)-dimensional space, so one dimension has been lost. [**?**]

The graph below visualizes the network structure. [Fig 7.3]

With skip-gram, the representation dimension decreases from the vocabulary size (V) to the length of the hidden layer (N). Furthermore, the vectors are more "meaningful" in terms of describing the relationship between words. The vectors obtained by subtracting two related words sometimes express a meaningful concept such as gender or verb tense, as shown in the following figure (dimensionality reduced). [Fig 7.4]

Continuous Bag of Words (CBOW)[3] is very similar to skip-gram, except that it swaps the input and output. The idea is that given a context, we want to know which word is most likely to appear in it.

The biggest difference between Skip-gram and CBOW is that the way the word vectors are generated. For CBOW, all the examples with the target word as target are fed into the networks, and taking the average of the extracted hidden layer. For example, assume we only have two sentences, "He is a nice guy" and "She is a wise queen". To compute the word representation for the word "a", we need to feed in these two examples, "He is nice guy", and "She is wise queen" into the Neural Network and take the average of the value in the hidden layer. Skip-gram only feed in the one and only one target word one-hot vector as input.

Figure 7.5 shows a 3D representation of word vectors, after a applying a nonlinear dimensionality reduction function like (t-SNE)[4][**?**]. The key to understand here is that having the vectors in 2 or 3 dimensions we can then move in some direction and find terms given by the contest. If we move through the male-female direction from the word

---

[3] `https://iksinc.online/tag/continuous-bag-of-words-cbow/`

[4] t-distributed Stochastic Neighbor Embedding

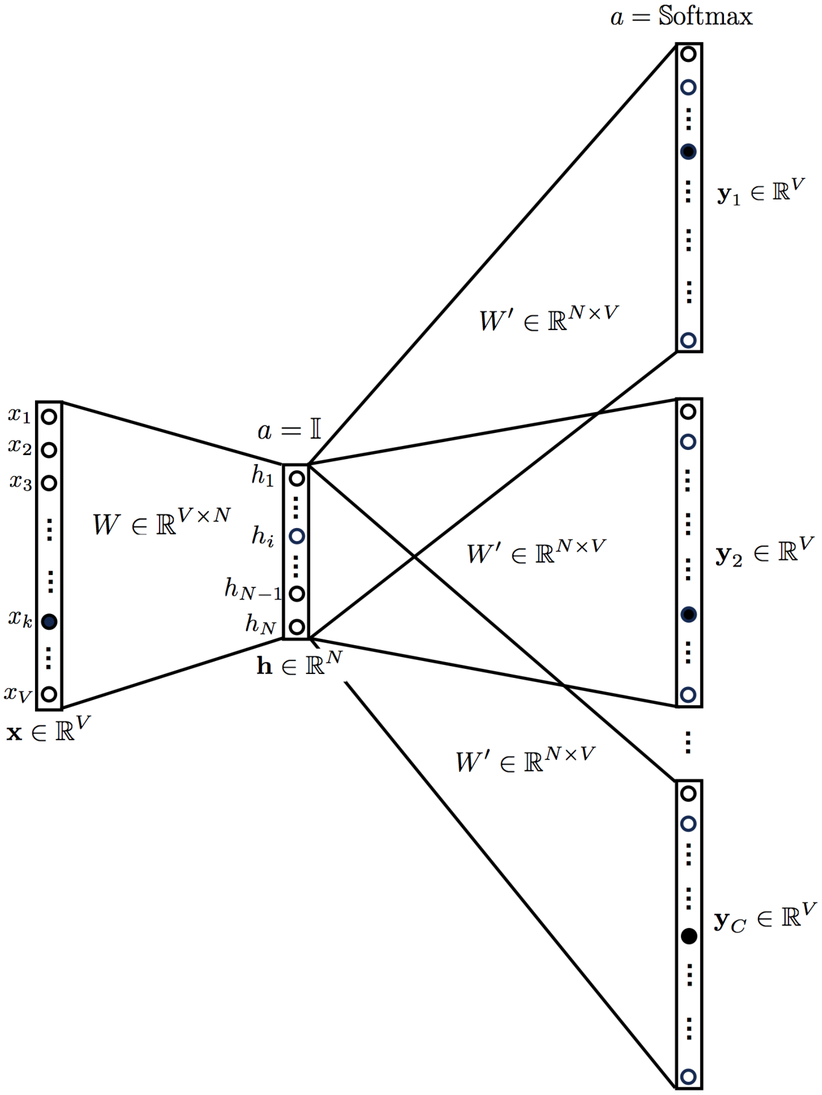

$a = \mathbb{S}\text{oftmax}$
$\mathbf{y}_1 \in \mathbb{R}^V$
$W' \in \mathbb{R}^{N \times V}$
x_1
x_2
x_3
$a = \mathbb{I}$
h_1
$W \in \mathbb{R}^{V \times N}$
h_i
$W' \in \mathbb{R}^{N \times V}$
$\mathbf{y}_2 \in \mathbb{R}^V$
h_{N-1}
x_k
h_N
$\mathbf{h} \in \mathbb{R}^N$
x_V
$\mathbf{x} \in \mathbb{R}^V$
$W' \in \mathbb{R}^{N \times V}$
$\mathbf{y}_C \in \mathbb{R}^V$

Figure 7.3: Skip-gram: The word embedding for the target words can be obtained by extracting hidden layers after feeding the one-hot representation of that word into the network.

vector representing the word *man*, we are likely to find the word *woman.* Likewise we are going to find the word *queen* if we start from the word vector representing the word *queen.*

It is claimed that Skip-gram tends to do better in rare words. Nevertheless, the performance of Skip-gram and CBOW are generally similar.

FastText is an extension to Word2Vec proposed by Facebook in 2016. Instead of feeding individual words into the Neural Network, FastText breaks words into several n-grams (sub-words). For instance, the tri-grams for the word apple is app, ppl, and ple (ignoring the starting and ending of boundaries of words). The word embedding vector for apple will be the sum of all these n-grams. After training the Neural Network, we will have word embeddings for all the n-grams given the training dataset. Rare words can now be properly represented since it is highly likely that some of their n-grams also

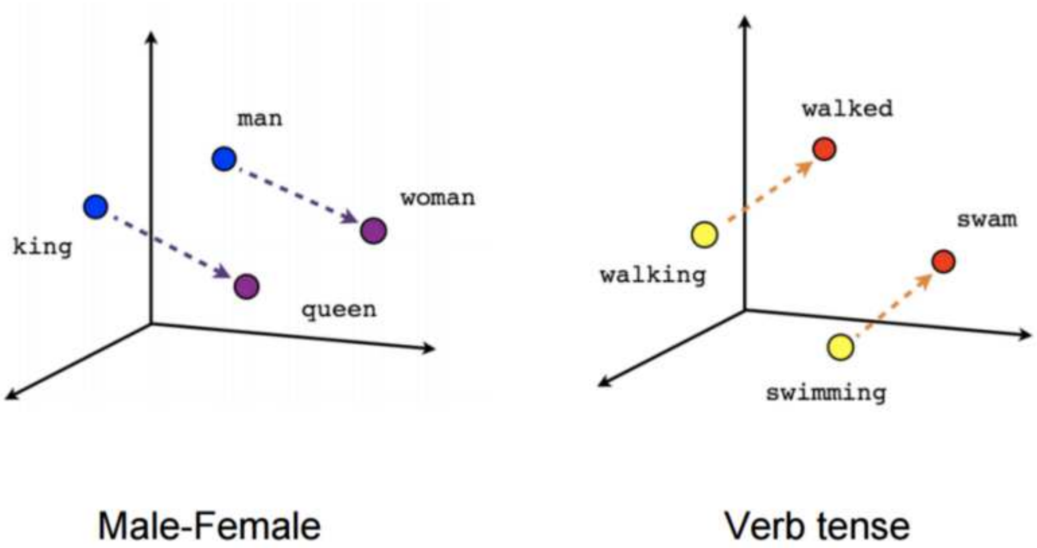


Figure 7.4: Skip-gram: the vectors are more "meaningful" in terms of describing the relationship between words.

appears in other words.[?]

Due to the success of word embeddings in a variety of NLP applications, some existing studies evaluate word embeddings in representing word semantics quantitatively. Most of them focus on evaluating the word embeddings generated by different approaches. **?** [?] presented the first systematic evaluation of word embeddings generated by four models, i.e., DISSECT, CBOW using word2vec, Distributional Memory model, and Collobert and Weston model using a corpus of 2.8 billion tokens in the general English domain. They tested these models on fourteen benchmark datasets in five categories, including semantic relatedness, synonym detection, concept categorization, selectional preferences, and analogy. They found that the word2vec model, CBOW, performed the best for almost all the tasks. **?** [?] trained the CBOW model of word2vec, C&W embeddings [?] , Hellinger PCA [?], GloVe [?], TSCCA [?], and Sparse Random Projections [?] on a 2008 GloVe dump, and tested on the same fourteen datasets. They found that the CBOW outperformed other embeddings on 10 datasets. They also conducted an extrinsic evaluation by using the embeddings as input features to two downstream tasks, namely noun phrase chunking and sentiment classification. They found the results of CBOW were also among the best. **?** [?] conducted a similar intrinsic evaluation, they additionally evaluated the skip-gram models of word2vec, CSLM word embeddings [?], dependency-based word embeddings, and combined word embeddings on four NLP tasks, including Part-Of-Speech tagging, chunking, named entity recognition, mention detection, and two linguistic tasks. They trained these word embeddings on the Gigaword corpus composed of 4 billion words and found that the dependency-based word embeddings gave the best performance on the NLP tasks and that the combination of embeddings yielded

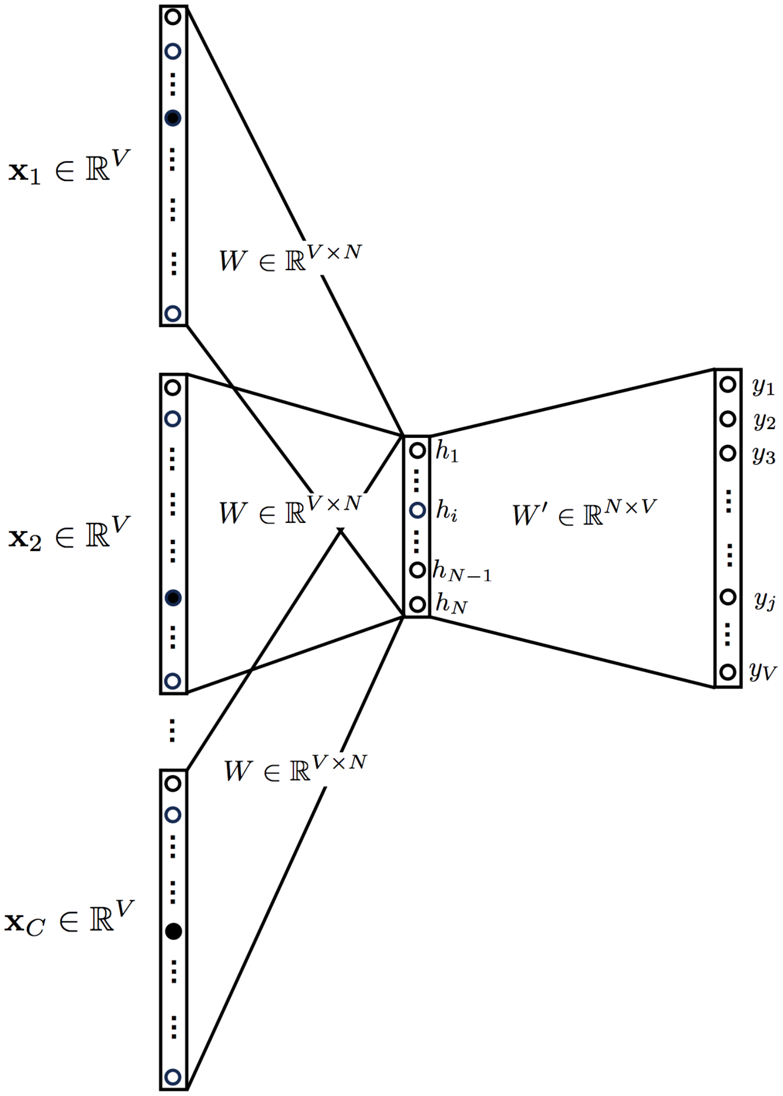

$\mathbf{x}_1 \in \mathbb{R}^V$
$\mathbf{x}_2 \in \mathbb{R}^V$
$\mathbf{x}_C \in \mathbb{R}^V$
$W \in \mathbb{R}^{V\times N}$
$W \in \mathbb{R}^{V\times N}$
$W \in \mathbb{R}^{V\times N}$
h_1
h_i
h_{N-1}
h_N
$W' \in \mathbb{R}^{N\times V}$
y_1
y_2
y_3
y_j
y_V

Figure 7.5: The main difference between CBOW and Skip-gram is that CBOW swaps the input and output.

significant improvement. However, few of these studies evaluated word embeddings for tasks in the biomedical domain. As most of the aforementioned studies evaluate word embeddings in the general (i.e., non-biomedical) NLP domain, only one recent study by **?** [**?**] evaluates word embeddings in the biomedical domain, to the best of our knowledge. They trained the CBOW model on two biomedical corpora, namely clinical notes and biomedical publications, and one general English corpora, namely GloVe. The word embeddings were evaluated on subsets of UMNSRS dataset, which consisted of pairs of medical terms with the similarity of each pair assessed by medical experts, and on a document retrieval task and a word sense disambiguation task. They found that the semantics captured by the embeddings computed from biomedical publications were on par with that from clinical notes.

There is a rich body of work on learning general-purpose word embeddings. So far, there are only very few studies focusing on learning domain-specific word embeddings. For example, ? ? [?] uses information from a disease lexicon to generate disease-specific word embeddings. The main objective is to bring in-domain words close to each other in the embedding space while pushing out-domain words away from in-domain words. ? ? [?] only concerns whether a word is in-domain or not. Another example is ? ? [?], describes a novel method to train domain-specific word embeddings from sparse texts. First, it proposes a general framework to encode diverse types of domain knowledge as text annotations; then, it develops a novel Word Annotation Embedding (WAE) algorithm to incorporate diverse types of text annotations in word embedding. Evaluating the method on two text corpora resulted in demonstrating the effectiveness of the method in learning domain-specific word embeddings.

The existing studies of automating ICD-9-CM code assignment can be classified into two groups. Through examining how professional coders assigning ICD-9-CM codes, the first one used rule-based approaches. ? ? [?] developed a rulebased system considering factors such as uncertainty, negation, synonymy, and lexical elements. ? ? [?] used Decision Tree (DT) and Maximum Entropy (ME) to automatically generate a rule-based coding system. ? ? [?] composed a hybrid system consisting of a machine learning system with natural language features, a rule-based system based on the overlap between the reports and code descriptions, and an automatic policy system. Their results showed better performance than each single system. The second group employed supervised machine learning methods for the assignment task, and their performance has been being equivalent or even better than those rule-based systems that need experts manually crafting knowledge. ? ? [?] used a stacked model to combine the results of four modules: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Pattern Matching (PM) and a hybrid Medical Text Indexer (MTI) system. ? ? [?] used ME and SVM classifiers, enhanced by a feature engineering module that explores the best combination of several types of features. ? ? [?] proposed a hierarchical text categorization method utilizing the ICD-9-CM codes structure. Along with the introduction of supervised methods, many past studies indicated that data imbalance problem can severely affect the classifier's performance. For example, ? ? [?] found that 874 of 1,231 ICD-9-CM codes in UKLarge dataset have less than 350 supporting data, whereas only 92 codes have more than 1,430 supporting data. The former group has macro F1 value of 51.3%, but the latter group only has 16.1%. To resolve data imbalance problem, they used optimal training set (OTS) selection approach to sample negative instance subset that provides best performance on validation set. However, OTS did not work on UKLarge dataset because several codes have so few training examples that even carefully selecting negative instances could not help. When ? ? [?] found that 85% of the whole death certificate dataset is associated with only top 20 common cancers, whereas the other 65 rarer cancers only have the rest 15% of the dataset, they tried to construct the balanced training set by randomly sampling a static number of negative examples for each class. Their results reflected the benefits of having more training data in improving the classifiers' performance. Since result of original model learned with imbalanced data is not provided, we cannot know the actual improvement. In addition, to deal with codes that only appear once in the dataset, **?** **?** [**?**] used a rule-based module to supplement ME and SVM classifiers. For the recent studies, **?** **?** [**?**] proposed an automatic feature extraction method, capturing semantic relational tuples. They proved the semantic relational tuple is able to capture information at semantic level and it contribute to ICD-9-CM classification task in two aspects, negation identification and feature generation. Another recent study to solve training data shortage problem, **?** **?** [**?**] proposed to strategically draw data from PubMed to enrich the training data when there is such need. The evaluation results indicate that their method can significantly improve the code assignment classifiers' performance at the macro-averaging level.

CHAPTER 8

CONCLUSION

In this thesis we proposed domain-specific word embeddings for ICD-9-CM classification. In particular, we showed how to collect the dataset and enlarge it with other medical data.

We gathered together 3 main corpora of specific medical data: the first one from the emergency room discharge records of the Forlì Hospital, where we collected more than 700k real anonymous diagnosis written by doctors when sending home patients, the second one has been downloaded from all the italian medical articles available on Wikipedia and the last one was the official ICD-9-CM dictionary of the more than 16k definitions of diagnosis and their corresponding code.

For the general and comparison models, as we weren't able to use news data (i.e. GoogleNews, the most popular) because they aren't provided in italian language, we were forced to use other pre-trained models in italian. We found 2 of them with different backgrounds: the first one is the wikipedia dump of all the pages in it, the other one is provided with TED speech corpus translated in italian.

Due to the completely different nature of the two general purpose datasets, they showed when testing most similar words that they responded in a completely different way because they aren't filled with medical specific words in their starting dataset.

Subsequently, with the help of gensim tool and Word2Vec we trained each corpus separately forming 3 separate models and one joining all the documents of each corpus. Every model has been trained with the same hyperparameters. We showed the results of the training of this dataset compared to the general purpose retrieved from TED corpus and general Wikipedia italian dump. Through most-similar words tool with the help of 2D plot, we tested the different models and showed how they respond.

We pointed out some of the most common feature typical in a medical text, such as typo errors and medical jargon. Medical terminology is made up of numerous Greek and/or Latin suffixes and prefixes. It describes body parts, functions, surgical procedures, and is used in medical reports. Every language has his own lingo and even every specific group of doctors has its own. When writing down an emergency room discharge records, doctors very often use abbreviations and sort of codes known specifically in the medical jargon to describe body parts and common words. This is done for plenty of reasons: first of all in Italy doctors still use quite a lot handwritten emergency room discharge records, but even typing on a keyboard doctors are known to be a little bit lazy.

From our evaluated models we looked for the meaning of main commons medical abbreviations and on the other hand we discovered how a word is commonly abbreviated by doctors.

The reader has to understand that this process is automatic and the model was unsupervised. Most similar words to a given word it is a result solely based on the context taken from the training dataset.

We finally proofed that our domain-specific word embeddings should be chosen over the general purpose. We presented a related work where we tried to automatically assign the codes corresponding to a textual diagnosis through a classifier. This classifier used our domain-specific word embedding for weighting words instead of using a general-purpose one and we noticed a real gain in terms of accuracy.

As we can clearly see from this last results, we can conclude that domain-specific word embeddings is more suitable for this task. Moreover the 'JoinData' model is undoubtedly the most accurate among the other word embeddings, mostly over the general purpose word embeddings.

As for what concerns the classification problem of ICD-9-CM it's still an hard task to solve, but we can unmistakably state that for word embedding of this type in a domain-specific context, such as the medical one filled with jargon, technique words and mostly typos, a domain-specific word embeddings performs better than a general purpose one.

BIBLIOGRAPHY

[1] Shlomo Argamon and Shlomo Levitan. Measuring the usefulness of function words for authorship attribution. In *Proceedings of the 2005 ACH/ALLC Conference*, pages 4–7, 2005.

[2] Harald Baayen, Hans van Halteren, Anneke Neijt, and Fiona Tweedie. An experiment in authorship attribution. In *6th JADT*, volume 1, pages 69–75. Citeseer, 2002.

[3] Cindy Chung and James W Pennebaker. The psychological functions of function words. *Social communication*, 1:343–359, 2007.

[4] Neal Fox, Omran Ehmoda, and Eugene Charniak. Statistical stylometrics and the marlowe-shakespeare authorship debate. *Proceedings of the Georgetown University Roundtable on Language and Linguistics (GURT), Washington, DC, USA*, 2012.

[5] Jack Grieve. Quantitative authorship attribution: An evaluation of techniques. *Literary and linguistic computing*, 22(3):251–270, 2007.

[6] Moshe Koppel, Jonathan Schler, and Kfir Zigdon. Determining an author's native language by mining a text for errors. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 624–628, 2005.

[7] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[8] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

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

[10] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north*

american chapter of the association for computational linguistics: Human language technologies, pages 746–751, 2013.

[11] Leonid A Mironovsky, Alexander V Nikitin, Nina N Reshetnikova, and Nikolay V Soloviev. Graphological analysis and identification of handwritten texts. In *Computer Vision in Control Systems-4*, pages 11–40. Springer, 2018.

[12] Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 93–102, 2015.

[13] Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556, 2009.

[14] Sean Stanko, Devin Lu, and Irving Hsu. Whose book is it anyway? using machine learning to identify the author of unknown texts. *Machine Learning Final Projects*, 2013.

[15] Chris van der Lee and Antal van den Bosch. Exploring lexical and syntactic features for language variety identification. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 190–199, 2017.

[16] Lili Yang, Chunping Li, Qiang Ding, and Li Li. Combining lexical and semantic features for short text classification. *Procedia Computer Science*, 22:78–86, 2013.

[17] Ying Zhao and Justin Zobel. Effective and scalable authorship attribution using function words. In *Asia Information Retrieval Symposium*, pages 174–189. Springer, 2005.

RINGRAZIAMENTI

Se 10 mesi fa mi avessero detto che oggi mi sarei laureato, non ci avrei mai creduto. Avevo completato il 40% del mio percorso universitario ed ero in bilico tra una pandemia mondiale ed un calo di motivazione personale. Se oggi scrivo questa ultima pagina della mia tesi, lo devo ad alcune persone in particolare:

Ringrazio il mio relatore, per aver creduto in me anche quando non lo facevo più nemmeno io.

Ringrazio Flavio *per l'enorme pazienza e la disponibilità durante gli ultimi mesi.*

Un ringraziamento speciale ai miei genitori, che non hanno mai smesso di credere in me e nel credere che ce la potessi comunque fare nonostante tutto.

Ringrazio il mio amico, socio e compagno di studi Alberto*, anche se sicuramente un "grazie" non può essere abbastanza. E' stato fonte di motivazione durante gli ultimi mesi e sempre pronto a farmi credere possibile questo incredibile traguardo, considerando da quanto lontano provenivamo e quanto poco tempo avessimo per concluderlo.*

Ringrazio anche Pietro *per il sostegno, i consigli e l'esperienza che ci ha permesso di superare più velocemente gli ultimi esami.*

Ringrazio Ossama *e* Luca*, amici e soci nel mondo imprenditoriale, che mi ha rubato tempo all'università ma che sicuramente mi ha insegnato tantissimo. E' bello lavorare con voi, grazie per avermi permesso di assentarmi qualche giornata per completare questo lavoro.*

Ringrazio anche Beatrice, Marco *e* Claudio*: anche se ci siamo visti di rado negli ultimi mesi, mi avete sempre stimolato e fatto ri-credere sempre di più nell'importanza di questo traguardo.*

Ringrazio Rebecca*, per avermi supportato (e sopportato) "no matter what", per i weekend passati davanti al pc e per le vacanze in giro per l'Italia (quando ancora si poteva).*

Ringrazio Billo *per esser stato la miglior distrazione che potessi avere negli ultimi 5 mesi.*

Infine ringrazio il corona virus, dapprima accolto come una catastrofe ma poi sempre di più come segno che anche quando alcune porte si chiudono... beh, altre si aprono!

APPENDIX A

CODE

A.1 Dataset estraction

A.1.1 RCV1

We used as parameter depth 2 and this list of categories: *Salute* (i.e. health in italian), *Medicina* (i.e. medicine in italian), *Procedure mediche* (i.e. medical procedures in italian), *Diagnostica medica* (i.e. medical diagnostics in italian) and *Specialità medica* (i.e. medical specialty in italian). The language of the result pages is by default ”it” (i.e. italian).

A.1.2 GDELT

To retrieve a single document, we parsed the HTML of every Wikipedia page with HTMLParser subclassed with a customized class MLStripper, that stripped away all HTML tags.

```
class MLStripper(HTMLParser):
    def __init__(self):
        super().__init__()
        self.reset()
        self.fed = []

    def handle_data(self, d):
        self.fed.append(d)

    def get_data(self):
        return ''.join(self.fed)
```

```
def strip_tags(html):
    s = MLStripper()
    s.feed(html)
    return s.get_data()
```

A.2 Model

A.2.1 Feature extraction

Code A.1 shows how to load the model from the pre-trained vectors.

Code Listing A.1: How to load Wikipedia pre-trained model

```
# word2vec model
from gensim.models import Word2Vec
model = Word2Vec.load('wiki_iter=5_algorithm=skipgram_window=10
    _size=300_neg-samples=10.m')
```

A.2.2 Train model

Code A.2 shows how to build the vocabulary for the model and train it.

Code Listing A.2: How to build vocabulary and train model

```
# build vocabulary and train model
model = gensim.models.Word2Vec(documents,
                               size=200,
                               window=10,
                               min_count=2,
                               workers=10)
model.train(documents,
            total_examples=len(documents),
            epochs=10)
```

A.2.3 Evaluation

Code A.3 shows how to plot words' vectors. We used pyplot from *matplotlib*[1] module and PCA from *sklearn.decomposition*[2] module.

---

[1] `https://matplotlib.org/api/pyplot_api.html`

[2] `http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`

Code Listing A.3: How to plot words' vectors

```
from matplotlib import pyplot
from sklearn.decomposition import PCA
def plot(model, words):
    X = model[model.wv.vocab]
    pca = PCA(n_components=2)
    result = pca.fit_transform(X)
    pyplot.scatter(result[:, 0], result[:, 1])
    for i, word in enumerate(words):
        pyplot.annotate((word[0], float(round(word[1],
          2))), xy=(result[i, 0], result[i, 1]))
    pyplot.show()
```

LIST OF FIGURES

2.1 PubMed text mining results 18
2.2 One Hot vector 20
2.3 Skip-gram evaluation 22
2.4 Skip-gram relationship between words 23
2.5 CBOW and Skip-gram 24

3.1 Number of documents for authors in RCV1 31
3.2 David Lawder vs Alexander Smith top 20 words 33
3.3 David Lawder WordCloud in RCV1 34
3.4 Alexander Smith WordCloud in RCV1 35
3.5 Vocabularity Richness RCV1 CCAT_10 37
3.6 Positivity and Negativity scores for RCV1 authors 41
3.7 Word2Vec 42

4.1 Diabetes - 10 most similar words plotted for JoinData model 48
4.2 Patient - 10 most similar words plotted for JoinData model 50
4.3 Top 10 most frequent words in emergency room discharge records 51
4.4 Top 10 most frequent words in Wikipedia health-specific corpus 51
4.5 Top 10 most frequent words in ICD-9-CM Dictionary 53
4.6 Artery - 10 most similar words plotted for TED Corpus model 55
4.7 Artery - 10 most similar words plotted for Wikipedia General Corpus model 56
4.8 Artery - 10 most similar words plotted for JoinData model 57
4.9 Artery - 10 most similar words plotted for dictionary corpus model . . . 58
4.10 Artery - 10 most similar words plotted for Wikipedia health-specific model 59
4.11 Artery - 10 most similar words plotted for Emergency room discharge records model 60

7.1 PubMed text mining results 66
7.2 One Hot vector 68
7.3 Skip-gram evaluation 70

7.4 Skip-gram relationship between words 71
7.5 CBOW and Skip-gram 72

List of Tables

3.1 Top 3 n-grams in RCV1 corpus 35
3.2 Comparing stylometric features of 2 authors in RCV1 37
3.3 Top 5 IFIDF n-grams in RCV1 corpus 39

4.1 First 5 diagnosis that contains at least one typo error beginning with the letter “a” 47
4.2 Most-similar words to: *emoragia* (typo of bleeding) and *ati* (typo of limbs) 48
4.3 Most similar words to: *tac* (CT scan), *radiografia* (radiography), *paziente* (patient) 49
4.4 Most similar words to: ‘*als*’ (acronym of Advanced Life Support) and ‘*pz*’ (abbreviation of patient) 49
4.5 First 5 ICD-9-CM coded diagnosis that contains the word *altre* (other) . 52
4.6 General purpose - Most similar words in TED corpus model. 54
4.7 General purpose - Most similar words in Wikipedia corpus model 54
4.8 Domain Specific - Most similar words in JoinData model 54
4.9 Domain Specific - Most similar words in dictionary corpus model 55
4.10 Domain Specific - Most similar words in Wikipedia health-specific corpus model 55
4.11 Domain Specific - Most similar words in emergency room discharge records corpus model 56
4.12 F1 Micro of each word embedding, both domain-specific and general purpose 58
4.13 F1 Weighted of each word embedding, both domain-specific and general purpose 59