Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

On Authorship Attribution

Relatore:
Chiar.mo Prof.
DANILO MONTESI

Presentata da:
Gabriele Calarota

Correlatore:
Dott.
FLAVIO BERTINI

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

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 9.4]

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 METHODS

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 9.1, 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

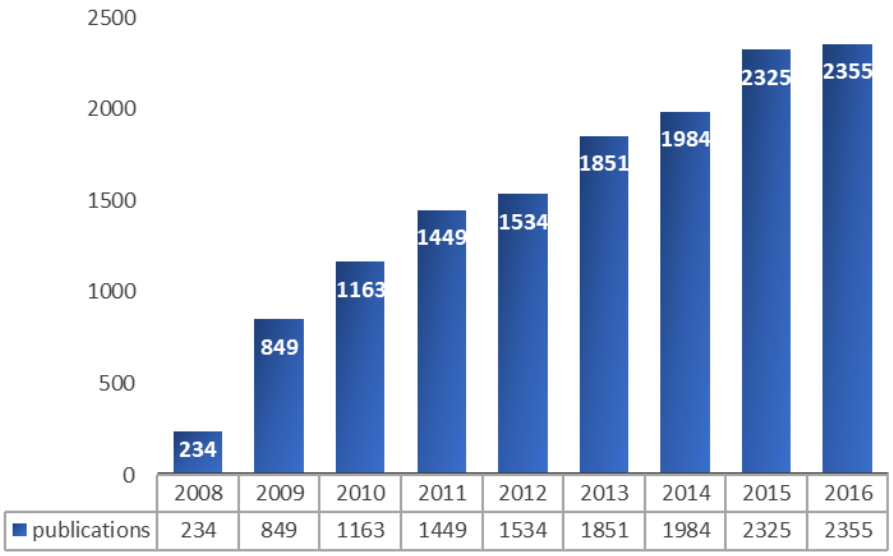

Figure 2.1: The number of publications in PubMed using the query word "*text mining*" or "*literature mining*" in the title or abstract over the last years. Search detail: *text mining* [Title/Abstract] or *literature mining* [Title/Abstract].

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

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

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.

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.

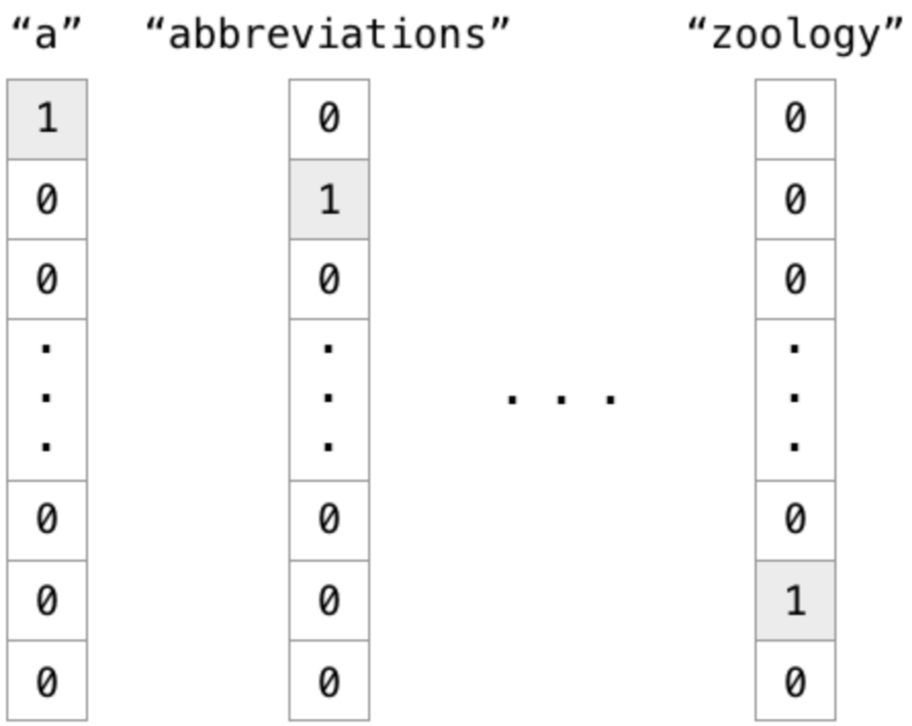

Figure 2.2: One Hot Vector: A simple and easy way to implement word vectors. 1 Bit set to 1 and all the others to 0. The dimension of the vector depends on the size of the vocabulary in input.

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 9.3]

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 9.4]

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

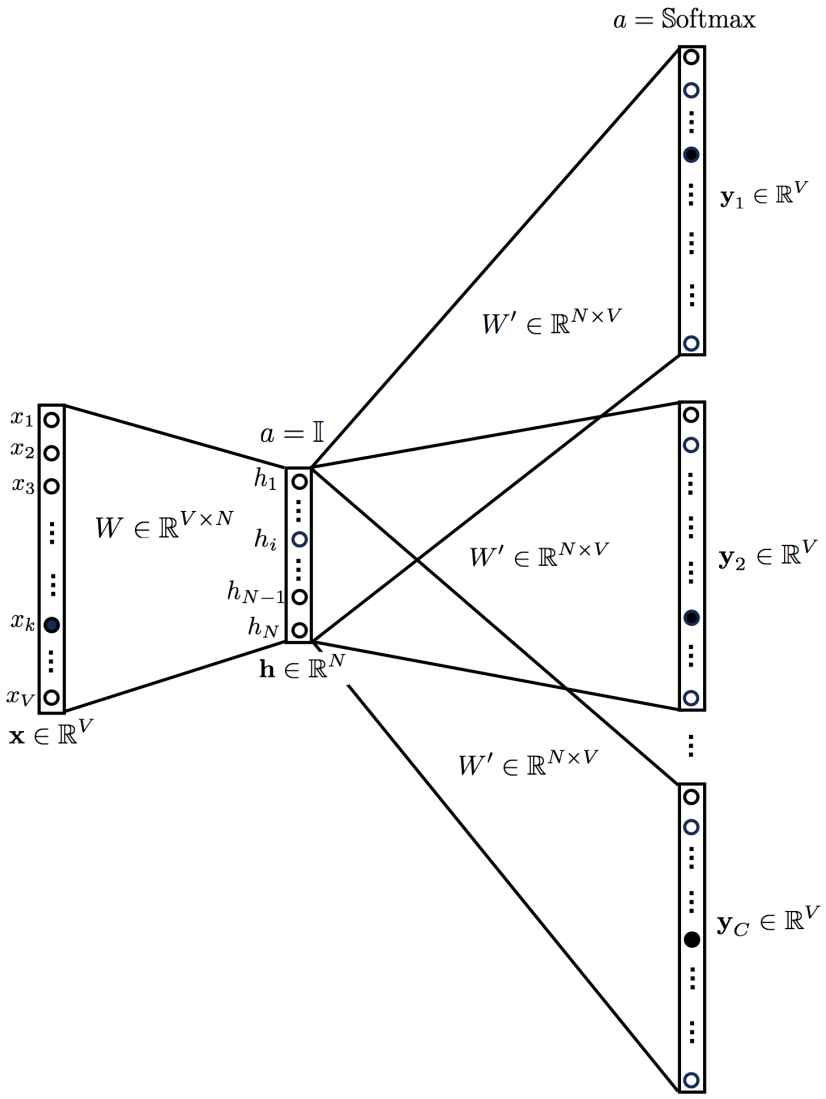

Figure 2.3: Skip-gram: The word embedding for the target words can be obtained by extracting hidden layers after feeding the one-hot representation of that word into the network.

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/`

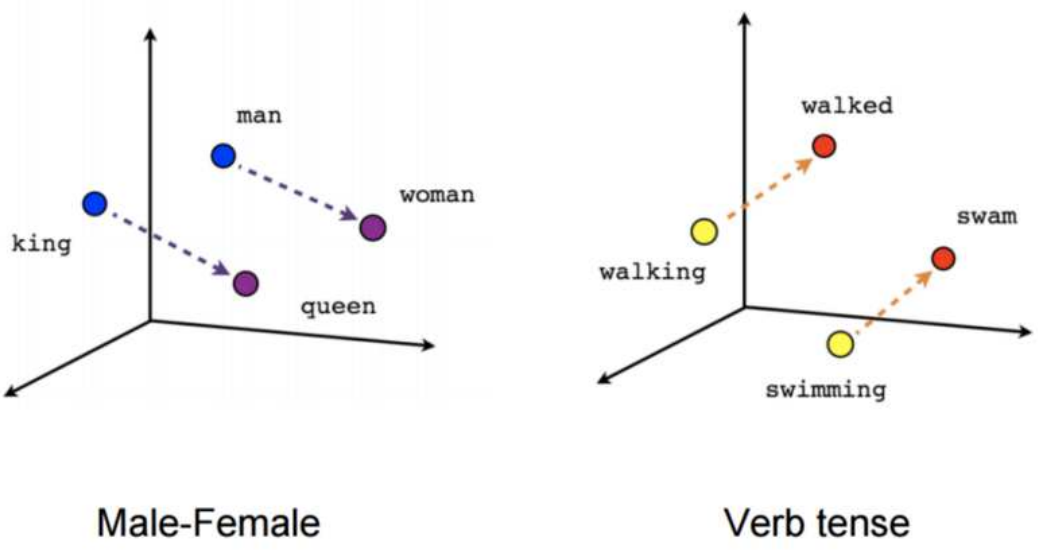

Figure 2.4: Skip-gram: the vectors are more "meaningful" in terms of describing the relationship between words.

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

Figure 9.5 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

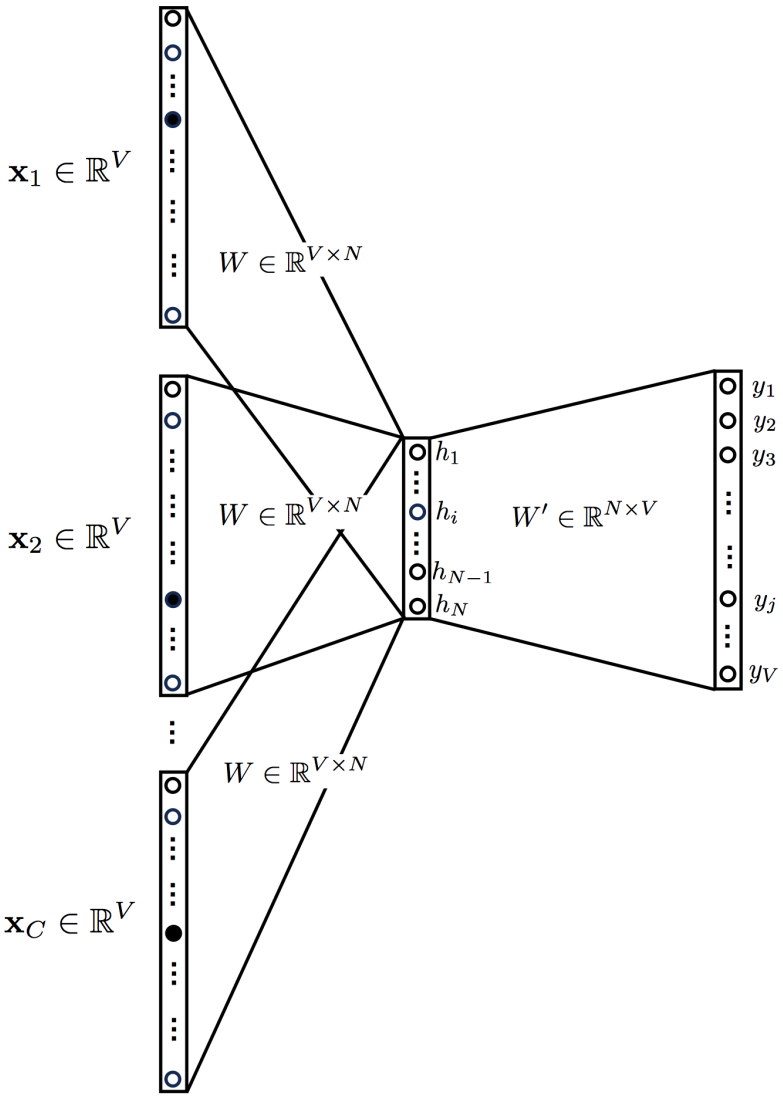

Figure 2.5: The main difference between CBOW and Skip-gram is that CBOW swaps the input and output.

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

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 in hand-written text, which was used for years in the past and it's still studied nowadays [5], 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 sequence of characters. Some of the character-level features are alphabetic characters count, digit characters count, uppercase and lowercase character counts, letter frequencies, character n-gram fre-quencies. This type of feature extraction techniques has been found quite useful to quantify the writing style.[3] 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 dimensionality of these approaches are vastly increased and it has a curse of dimensionality problem. A simple way of explaining what a character n-grams could be with the following example: assume that a word "student" is going to be represented by 2-gram characters. So, the resulting sets of points will be "st", "tu", "ud", "de", "en", "nt".

In [6] 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, Punctuaction 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.

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 [2], [1], [7]. 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. 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. Since there are uniquely 10, 000 words in the whole dataset you can choose the range of the words you want to use in your feature vector. We have also shown how to extract top words usage author-wise in every author's text corpus. As expected top words are determiners that every writer use while constructing an English sentence. For

example, for Arthur Conan Doyle top 20 words are "the, to, of, he, a, and, i, that, in, you, was, it, she, his, her, had, as, not, with, for" but for Charles Dickens they are "the, to, of, i, and, a, in, that, it, is, he, she, be, her, you, was, as, not, with, for" in decreasing order. Even though the two sets are mostly the same the orders 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, the content and word usages for every book can be seen as a picture of some forms as illustrated for two sample books in the 50-author dataset in for Charles Dickens's famous book Oliver Twist and Mark Twain's book Horse Tale. Afterwards, stop words (common words) are removed and using the frequency of remaining words we have plotted it on a boy's figure and a horse's figure, respectively in using word-cloud library here.
In both figures, most frequent words are written in bigger fonts and these words are recognized firstly when looking at the figures.

The same methodology can be applied to 3-author dataset as well. Using Vectorizer 3 a simple model is being shown on the tutorial. It simply takes the whole corpus and vectorize it based on the frequency of each word. It is found that in the training set, there are 25068 unique words whereas in the testing set the number of unique words is 17546. It is also shown how to extract total number of appearance in the whole corpus. As an example a search has been done on "Frankenstein" and it is found to appear "6301" times. In order to apply the same methodology to 50-author dataset a conversion algorithm has been provided for readers.

One important aspect of the authors in our dataset is that they are English language writers. However, some of these authors like Charles Dickens or William Black are originally from United Kingdom and some like Thomas Nelson Page are from United States. There are slightly differences between British English and American English. These differences also affect the usage of some of the words in the author's work. Words in American English such as "humor, color, honor, endeavor, theater" are used as "humour, colour, honour, endeavour, theatre" in British English. Using these features could improve the accuracy of the model and a simple algorithm has been provided to use it with Word2Vec here.

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 subsequence of length N of some sequence of sentence while a Skip-gram is a N-length subsequence where the components occur at a distance of at most k from each other [4]. In order to extract N-grams from a given text data a model has been built and tested on 50-author and 3-author dataset 6 . Stop-words have not been considered while constructing the N-grams. In 3-author dataset, for Edgar Allan Poe “upon, let us, general john b, general john b c” are the most occurrent uni-gram, bi-gram, three-gram, and four-gram respectively whilst in Mary Shelley they are “ one, lord raymond, let us go, nearest town took post” and in HP Lovecraft they are “one, old man, heh heh heh, oonai city lutes dancing”. Features such as “lord raymond” or “heh heh heh” could be a good identifiers as one mentions about the specific character in the book whilst the other one shows a preference of the author when it comes to phrasing an emotion.

Table 4.1 contains uni-grams, bi-grams, three-grams from authors Arthur Conan Doyle, William Carleton, Anne Manning. It also provides characteristic features about the persona in their books such as “sugar princess, mr george, sir john”. Same task can be also be achieved with considering the stop-words and can be better visualized as in Figure 4.3. In this task we have simply scanned through all consecutive word pairs of two and saved it in a dictionary. After sorting the dictionary based on the frequency of each bi-grams we have plotted it on a network diagram.

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 [31]. In order to apply this feature to both 50-author dataset and 3-author dataset a model has been built here. For 3-author dataset due to the large number of texts of Edgar Allan Poe, it was dominating the overall richness number but when normalized with the overall text number for each author the value became around 0.9. As for 50-author dataset the vocabulary richness has been found the lowest for William Carleton and the highest for Henry Haggard. Overall distribution of vocabulary richness is plotted in Figure 4.4.

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. During the preprocessing of 50-author dataset we have excluded punctuations from the dataset. Hence, calculating stylometric features are not available at the current stage for 50-author dataset. However, we can compute these features for 3-author dataset as in here. Overall distribution of some of the features introduced here are applied and the resulting density measures are calculated for each author and shown in Table 4.2. Among these 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.

3.2.5 Power Words

These are the words that affect the readers emotionally. It could be shown under the category of "semantic features", too. These words give emotional excitements or fears and serve as an emotional roller coaster. A great example could be given from the Winston Churchill. "We have before us an ordeal of the most grievous kind. We have before us many, many long months of struggle and of suffering. You ask, what is our policy? I can say: It is to wage war, by sea, land and air, with all our might and with all the strength that God can give us; to wage war against a monstrous tyranny, never surpassed in the dark, lamentable catalog of human crime. That is our policy. You ask, what is our aim? I can answer in one word: It is victory, victory at all costs, victory in spite of all terror, victory, however long and hard the road may be; for without victory, there is no survival." These frequency of such words in the database can also be studied to see how the authors are affectively using such words. The power words that are in 50-author dataset is also provided and their frequency for every author has been calculated. Then their density value which is the total number of power words divided by the text fragments for every author has been recorded. Figure 4.5 shows the distribution of these recordings for every author. The writings of Jane Austen has the most power words density whilst William Carleton has the lowest usage of such words.

3.2.6 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. In order to implement the use case of function words in our dataset a model is built. The list of commonly used function words are chosen from

Robert Layton's book on data mining [32] and the overall function word density measure is plotted on Figure 4.6. Among the authorsin training set, George Moore has the lowest usage of function words whereas Bret Harte has the highest function words density.

3.2.7 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 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 [31]. By considering every authors texts alone within the text corpus a Tf-Idf model has been built both for 3 and 50-author datasets. In the model, not only the single forms of word tokens but their n-grams are considered as well. As for the 3-author dataset, "afforded means, aware fact, dungeon make, perfectly uniform wall" are found to have highest Tf-Idf scores for Edgar Allan Poe, "fumbling mere mistake, occurred fumbling, mistake, fumbling" for HP Lovecraft, and "looked, beneath speckled, cheering fair, cottages wealthier, counties spread, happy cottages wealthier, lovely spring" for Mary Shelley. Table 4.3 provides the top 10 words and n-grams with highest Tf-Idf scores for the 50-author dataset. Comparing between Table 4.1, 4.3 new meaningful words have appeared that could serve as a new feature for each author such as "row, canal, passenger" for A. Doyle, or "writer, virtue, tale" for H. Greeley.

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 [31]. 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 [34]. 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 [34]. Some of these features can also be implemented by using 3-author or 50-author dataset. By making use of the part of speech tagging, a model has been built to analyze the usage of adjectives, nouns, and verbs in every author's training text corpus 15 . The main steps of the model consists of tokenizing text pieces author-wise and searching through the adjective, noun, verb list among the tokens. The measure of density is being defined as number of searched element divided by the total number of tokens for every author. In 3-author dataset, it is found that Edgar Allan Poe's preferred sets of top 3 adjectives, verbs, nouns list "little, other, more, was, is, had, time man day", HP Lovecraft uses "old, great, many, was, had, were, man, night, time", and Mary Shelley's set is "own, other, many, was, had, be, life, heart, Raymond".

Same methodology has also been implemented on the 50-author dataset. Figure 4.8 shows the density measure comparison across authors who are A.Doyle, C.Darwin, C. Dickens, E. Wharton, H. Greeley. List of top five adjectives, verbs, and nouns used by every author has also been recorded in Table 4.5. Noun variations across different authors is much significant than adjective and verb variations as expected.

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 [35]. 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 [35]. 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 [25]. 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 model has been built. In the algorithm, the sentences that have positive polarity score have been labeled as positive and the negative polarity scored ones are labeled as negative. In 3-author dataset, the most negative person has been found to be HP Lovecraft whereas the most positive one is Mary Shelley. In the 50-author training set, again the works of the authors A. Doyle, C. Darwin, C. Dickens, E. Wharton, H. Greeley have been chosen and their polarity scores have been calculated. Figure 4.9 shows the calculated positivity and negativity index for the chosen authors. Among these five authors the most negative one is found to be E. Wharton and the most positive one is C. Darwin.

3.4.2 Synonym Usage

The preference to use synonyms and antonyms in different text structure could be an identifiable feature in different tasks as well. However, extracting such features and modeling it could not be an easy task. The simple approach could be creating a domain knowledge where the pairs of synonyms and antonyms are paired. Then, a simple brute force approach can be used to find such words within a specific window size of sentences. Another approach is to employ the word vectors and represent a sentence with the average of all word vectors in the sentence. Using these two approaches an example model has been created. In the example model, given a synonym. its antonym set can be retrieved using NLTK wordnet library. Also, a similarity score can be calculated comparing the average vector forms of two sentences. Synonym and antonym word usage in 3 and 50-author dataset has not yet been studied and applied. Interested researchers can use the example model to further explore their affect in the authorship attribution problems.

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 char- acterized 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 [31]. Word2Vec can still be implemented in such domains as well as in 3 and 50-author dataset, but the way to use such vector forms depend on the creativity of one's approach.

3.5.1 Vector embeddings of words (Word2Vec)

It gives the ability to represent a word in a vector dimension of your choosing. The ways to make use of Word2Vec in 3-author and 50-author 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. We built a model based on profile based Word2Vec training and using TSNE to decrease it to two dimensions. In the model, our baseline approach is to extract A. Doyle and E. Wharton's text data and train Word2Vec on both of these authors text sets separately. Then, we have checked the word closeness for "listen" in both of these authors using 300 dimensional word vectors. Figure 4.10 shows the closest words in 2 dimensions for E. Wharton. The same comparison can also be done between pre-trained word vectors of Google or 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. The measured cosine distance for A. Doyle and E. Wharton regarding the usage of "listen" is 0.094. In order to apply this strategy on sentence level, we can have a few ways to do so. One way is by simply taking the scaled average of Word2Vec vectors in the sentence. Another one is to employ Tf-Idf score of each word as a gain when calculating a sentence vector. We then take the scaled average of all word vectors in the text piece. For the simplicity, we only consider taking the average vector without Tf-Idf gain for now. In this case, "her lips were parted" has been compared for both A. Doyle and E. Wharton. The cosine distance has been recorded as 0.258 which is much larger than the distance for the word "listen". The reason is that "her lips were parted" is an exact phrase that is extracted from A. Doyle whereas "listen" is a common verb for both authors. The same comparison can also be done by considering the Google's pretrained Word2Vec. The cosine distance for "listen" between A. Doyle and pretrained set is found as 0.015 whereas for E. Wharton, it is 0.012. As for "her lips were parted", the cosine difference for A. Doyle and pretrained set is -0.009, and for E. Wharton, it is 0.012. This implies that E. Wharton uses "listen" close to the pretrained set which was trained on large corpus of text data. As for sentence comparison, the sentence average vector is closer to A. Doyle stating that this sentence is more likely to be written by A. Doyle then E. Wharton. The way to achieve this comparison criteria has been provided here for readers to move forward with this methodology.

3.5.2 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.

It aims to represent words in terms of fixed length, continuous and dense feature vectors. A very popular model architecture for learning distributed word vector representations (Word2Vec) using a neural network was proposed in Mikolov et al. (2013a, b). This technique captures semantic and syntactic word relations: Similar words are close to each other in the vector space. For example, it was shown in Mikolov et al. (2013c) that $vector[King] - vector[Man] + vector[Woman]$ results in the vector that is closest to the representation of the $vector[Queen]$. Another two well-known continuous representations of words are latent semantic analysis (LSA) (Wiemer-Hastings et al. 2004) and latent Dirichlet allocation (LDA) (Trejo et al. 2015). Unlike LSA and LDA, the vector representations obtained by Mikolov et al. (2013a, b) preserve linear regularities between words. Distributed representations can be extended to model not only words, but also larger language structures like phrases, sentences and documents (Le and Mikolov 2014).

chapter 4

State of the art

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

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.

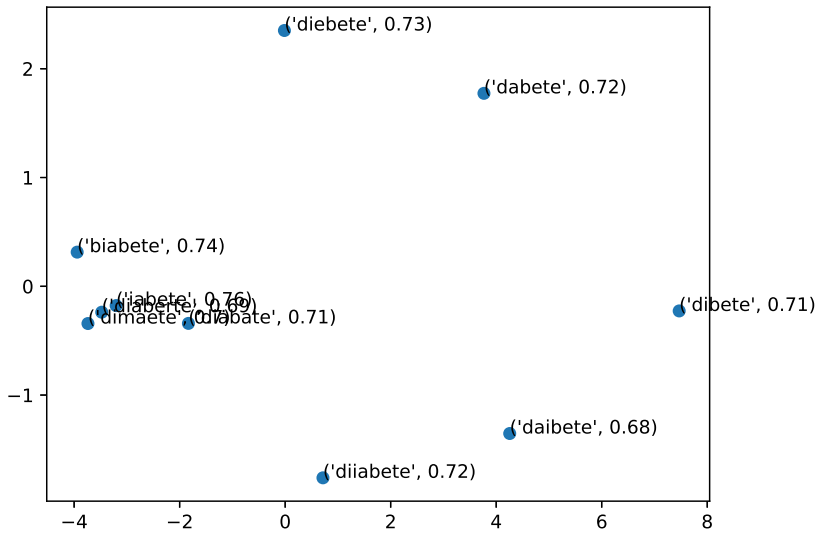

Figure 4.1: Top 10 most similar words to diabetes in the JoinData model.

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)

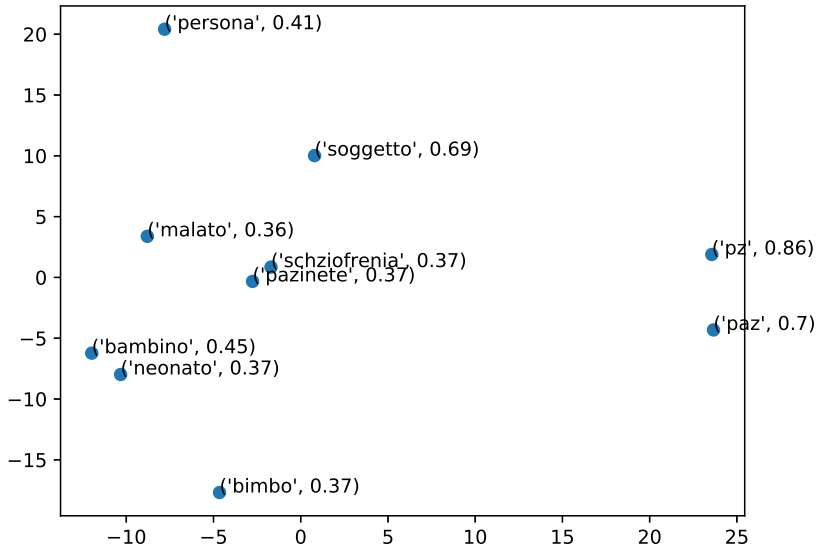

Figure 4.2: Top 10 most similar words to *paziente* (patient) in the JoinData model.

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

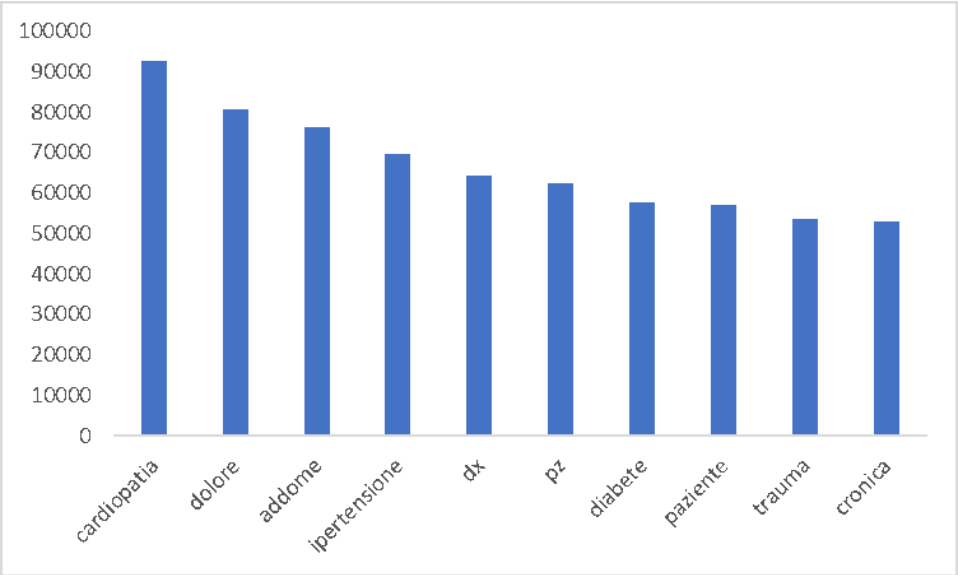

Figure 4.3: Top 10 most frequent words in the emergency room discharge records corpus.

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.

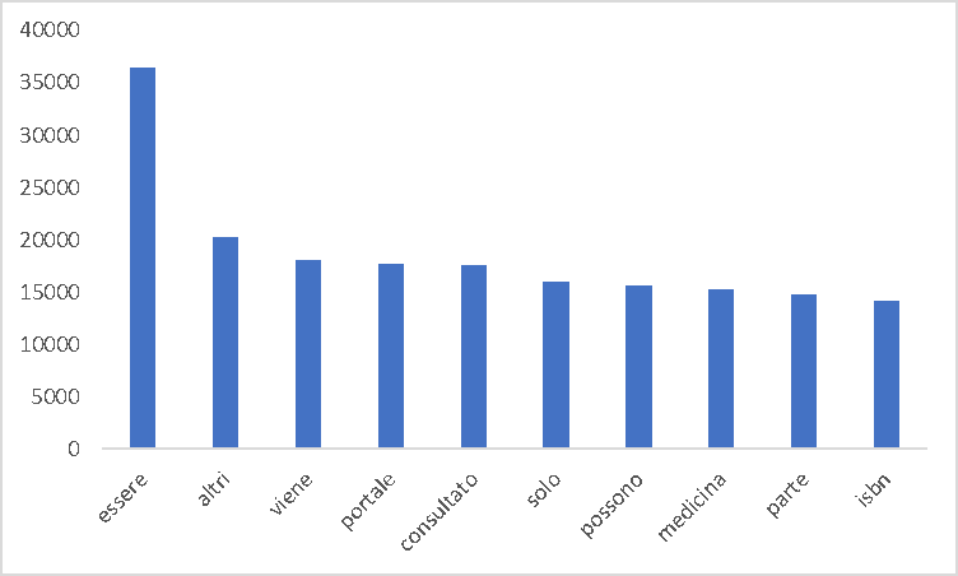

Figure 4.4: Top 10 most frequent words in the Wikipedia health-related corpus.

[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

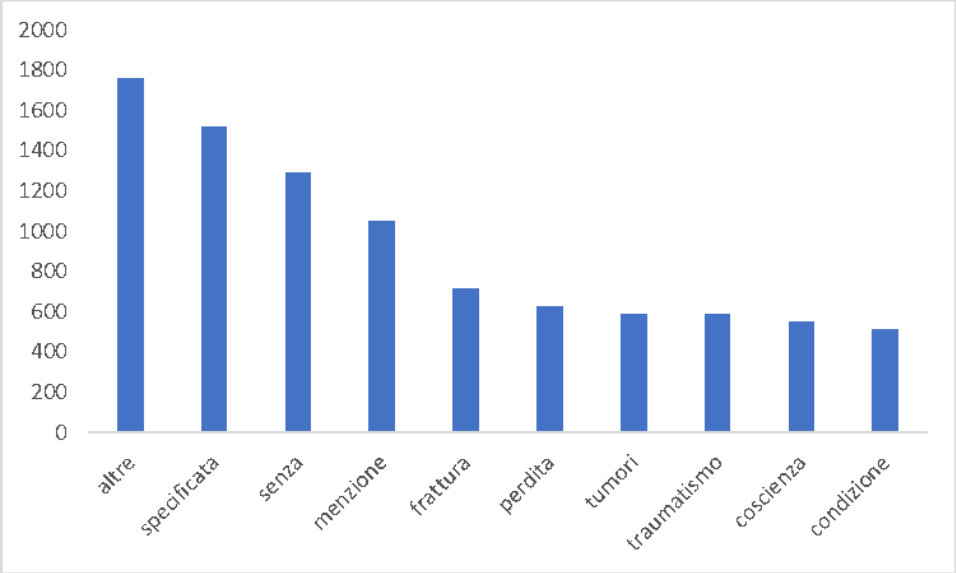

Figure 4.5: Top 10 most frequent words in the dictionary corpus.

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.

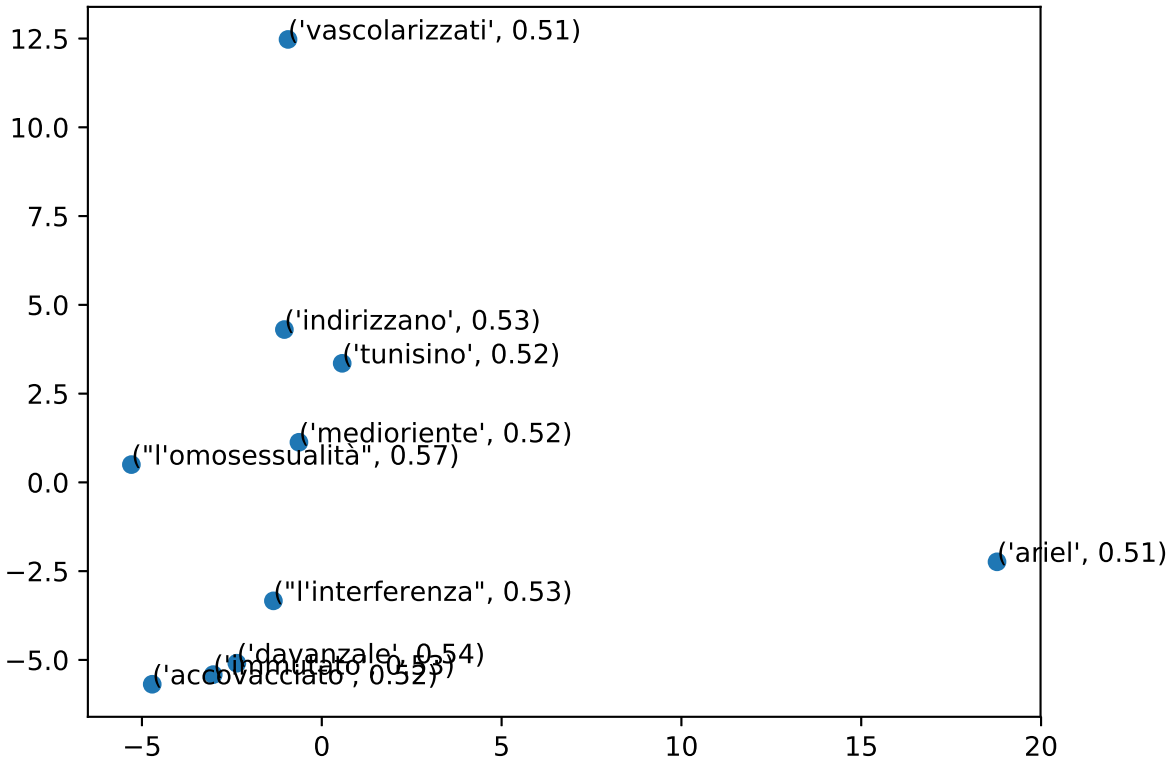

Figure 4.6: Plotting 10 most similar word vectors to *arteria* (artery) in the Ted Corpus model.

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

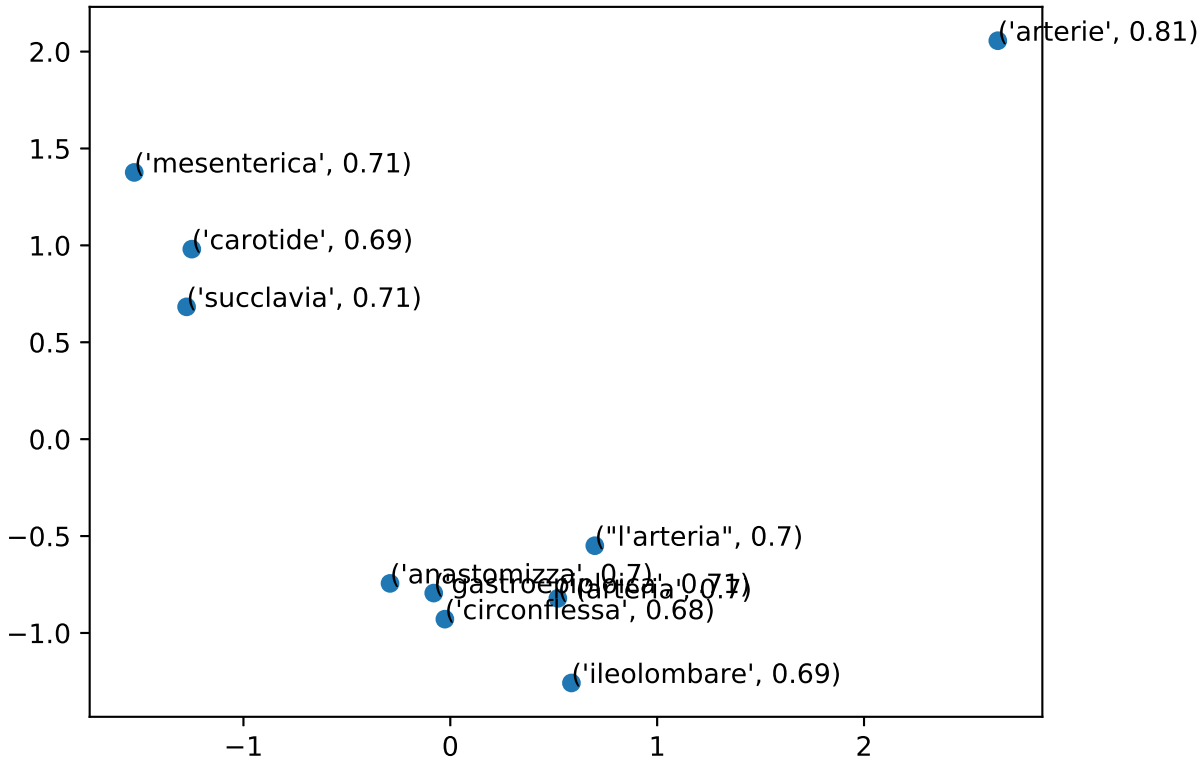

Figure 4.7: Plotting 10 most similar word vectors to *arteria* (artery) in the general Wikipedia corpus model.

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

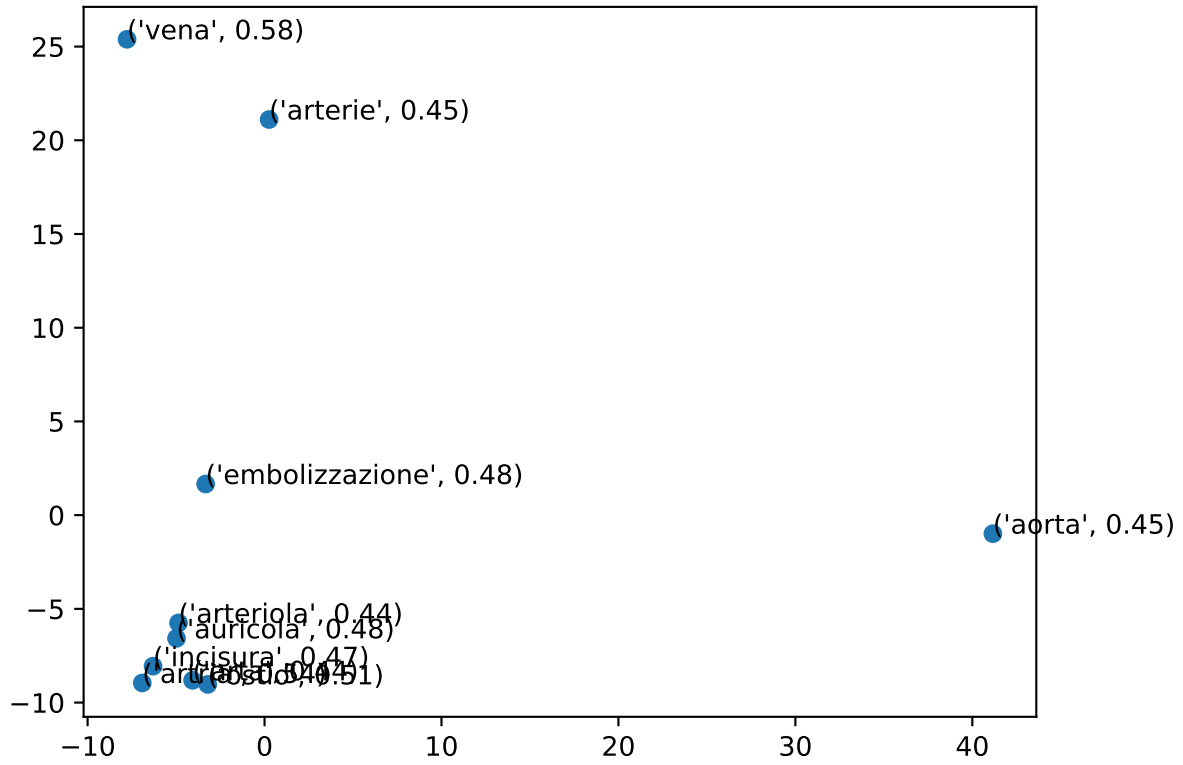

Figure 4.8: Plotting 10 most similar word vectors to *arteria* (artery) in the JoinData model.

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

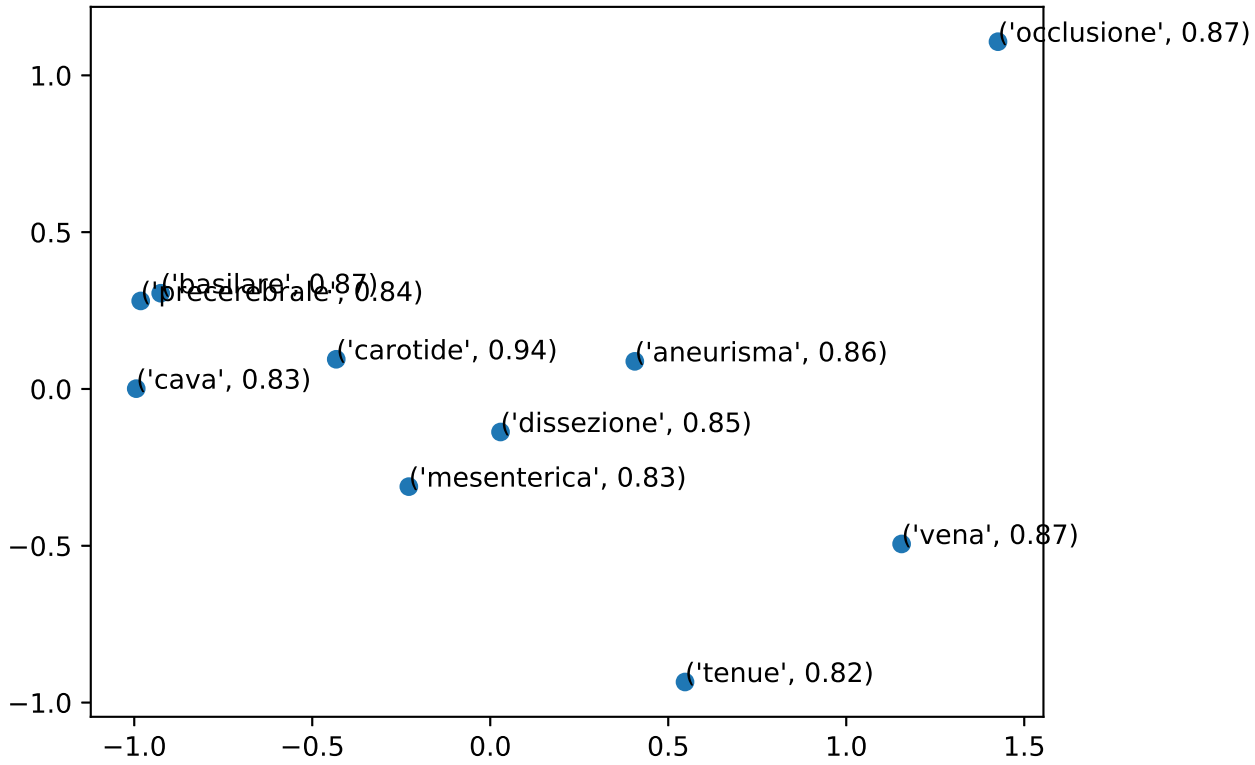

Figure 4.9: Plotting 10 most similar word vectors to *arteria* (artery) in the Dictionary data model.

- **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.

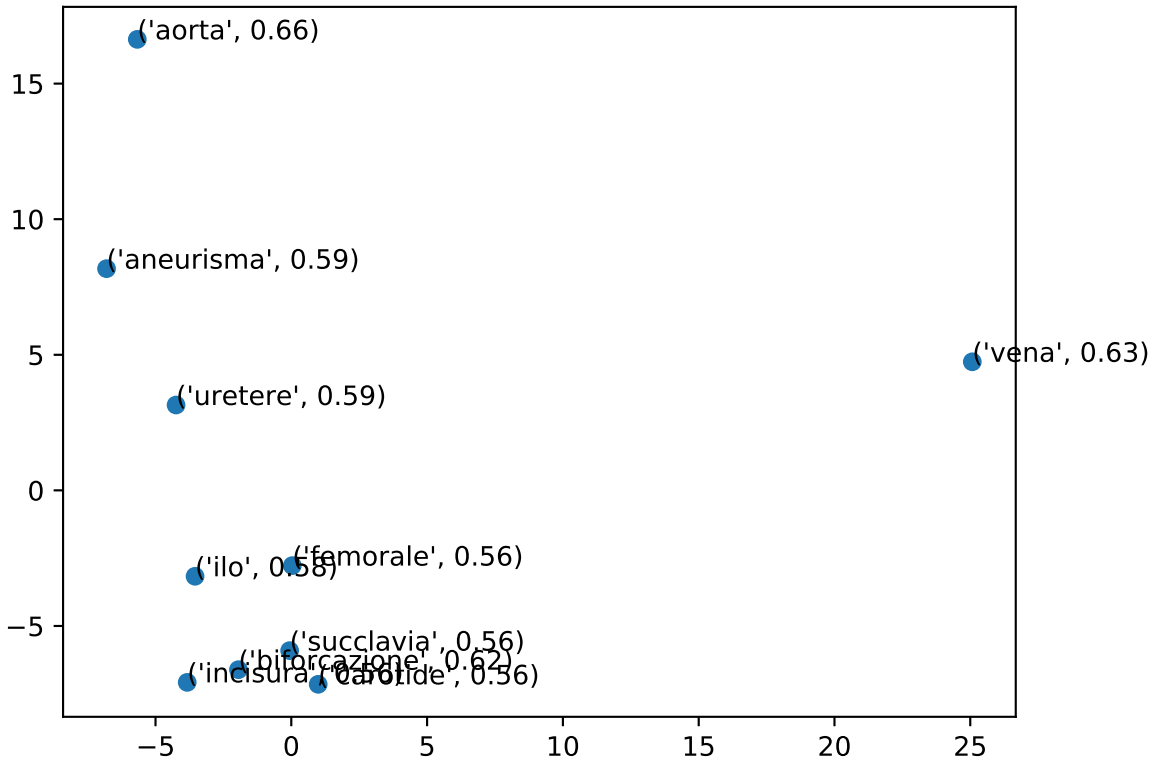

Figure 4.10: Plotting 10 most similar word vectors to *arteria* (artery) in the Wikipedia health-specific data model.

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.

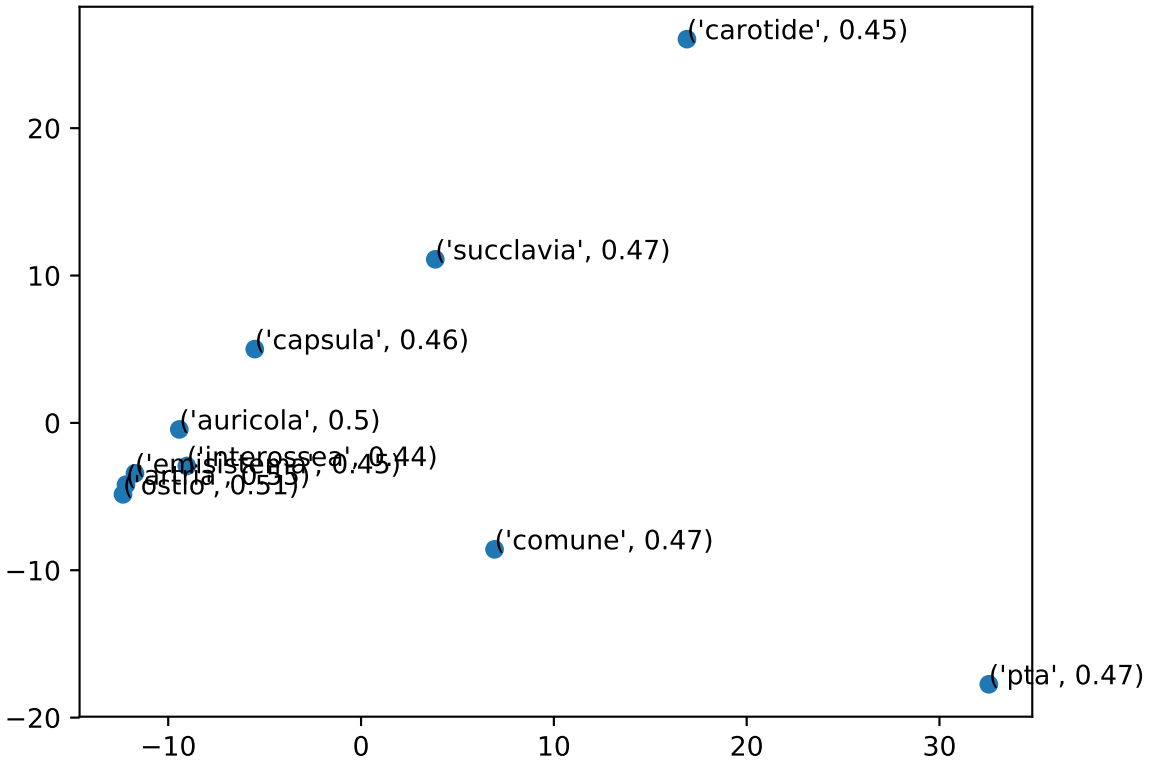

Figure 4.11: Plotting 10 most similar word vectors to *arteria* (artery) in the emergency room discharge records corpus model.

CHAPTER 5

Methods

5.1 Bag of Words

5.2 TFIDF

5.3 Doc2Vec?

5.4 SVM

chapter 6

Datasets selection

The dataset used for the ICD-9-CM is a result of a mixture of 3 main datasets: emergency room discharge records, Wikipedia medical pages and ICD-9-CM vocabulary.
We first collected more than 700 thousands real emergency room diagnosis of an Hospital in Italy. As this dataset is still too small to apply word embedding, we enlarged the dataset with more medical data founded in Wikipedia health-related pages, providing some useful technical terms to our dataset. In the end we decided to add the correct definition of each codified diagnosis in ICD-9-CM adding its vocabulary.
For the general and comparison models, as we weren't able to use news data (i.e. Google-News, 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, we expect that they will behave to the test in different way, in order to have a better point of view when testing our models.

6.1 GDELT

We collected 705'429 diagnosis from Forlì Emergency Room Hospital. Each line of data consist of an emergency room discharge record made by: anamnesis (ancient, upcoming, pharmacological and familiar), objective examination, clinical course.

- **Anamnesis**: cognitive investigation on the physiological and pathological precedents, individual and familiar, of a patient, written by doctors and aimed to the diagnosis.
- **Ancient pathological anamnesis**: the chronological description of every morbid

events that occurred during the patient's life with the sole exclusion of the pathology responsible for the hospitalization, which will instead be the subject of the upcoming pathological anamnesis.

- **Upcoming pathological anamnesis**: contains the story that patient tells and brought him to the hospital. It must be detailed because it allows doctors to know well the onset of the morbid event.
- **Pharmacological anamnesis**: a list of every prescribed or self-administered medicines (over-the-counter medications).
- **Familiar anamnesis**: consists of a collection of informations about the health status of parents and siblings.
- **Objective examination**: examination of the patient conducted by the doctor. It's called '*objective*' as it refers to the search for objective signs (other than subjective symptoms reported by the patient) indicative of a morbid state. It's the first step in the formulation of the diagnosis and the setting of a therapy.
- **Clinical course**: a description of how the disease behaves over time.

6.2 RCV1

As for the main corpus of data (12'437 documents and 13'195'758 words), the method for retrieving Wikipedia corpus data has been specifically designed to get only data in a domain-specific context, such as the medical. The process can be divided into two main parts: retrieving all the wikipedia pages in italian regarding medical topic and the dump of such pages.

6.3 The Guardian

We used the petscan software[1] to find all the relative pages to some medical category. We choose from the official categories list of Wikipedia[2] the following category: health, medicine, medical procedures, medical diagnosis and medical specialty.

[1] PetScan (previously CatScan) is an external tool that searches an article category (and its subcategories) according to specified criteria to find articles, stubs, images, and categories.
The tool can be found at: `https://petscan.wmflabs.org/`

[2] `https://en.wikipedia.org/wiki/Special:Categories`

6.4 Amazon Food Reviews

The language of the page had to be in italian and we chose 2 as the depth[3] parameter. [See A.1.1]

With the list of all the Wikipedia pages about medical topic, we queried the MediaWiki API[4] to get all the contents. The parameters used were:

- action = *mobileview*
  Action parameter tells which action to perform. *Mobileview* returns data needed for mobile views which consists of lighter data, with no picture and basically only text.

- format = *json*
  The output data in JSON format.

- page = *Title-of-the-page*
  Title of the page to process. It is a required parameter.

- sections = *0-*
  Pipe-separated list of section numbers for which to return text. "all" can be used to return for all. Ranges in format "1-4" mean get sections 1,2,3,4. Ranges without second number, e.g. "1-" means get all until the end.

- prop = *text*
  Which information to get. *Text* means only HTML of selected sections.

Example of the query with the page '*Epistassi*'(Nosebleed):
`https://it.wikipedia.org/w/api.php?action=mobileview&format=json&page=Epistassi&sections=0-&prop=text`

The response is a *json* list of the HTML text from section 0 till the end. To retrieve a single document, we then parsed the HTML with HTMLParser[5] and stripped away all the HTML tags with a customized class MLStripper in order to get Wikipedia data in plain text format. [See A.1.2]

The International Classification of Diseases, Ninth Revision, Clinical Modification (ICD-9-CM) is the U.S. health system's adaptation of international ICD-9-CM standard list of six-character alphanumeric codes to describe diagnoses. Standardizing codes improves consistency among physicians in recording patient symptoms and diagnoses for

[3]Depth of the category trees to search. 0 means to not use subcategories.

[4]English version: `https://en.wikipedia.org/w/api.php`

[5]An HTMLParser instance is fed with HTML data and calls handler methods when start tags, end tags, text, comments, and other markup elements are encountered. The user should subclass HTMLParser and override its methods to implement the desired behavior. For more information see official documentation at: `https://docs.python.org/2/library/htmlparser.html`

the purposes of payer claims reimbursement and clinical research. ICD-9-CM contains a list of codes corresponding to diagnoses and procedures recorded in conjunction with hospital care. These codes may be entered onto a patient's electronic health record and used for diagnostic, billing and reporting purposes. Related information also classified and codified in the system includes symptoms, patient complaints, causes of injury, and mental disorders.

The ICD-9-CM Classification, in the Italian translation prepared and published by the ISTAT, Classification of diseases, traumas and causes of death (9 revision, 1975), has been used, in accordance with the Decree of Ministry of Health of July 26, 1993, for the codification of clinical information detected through the emergency room discharge record (i.e. in italian *'Scheda di Dimissione Ospedaliera', SDO*). With the ministerial decree n. 380 of 20 October 2000 the codification of health information of the SDOs is carried out with the classification ICD-9-CM version 1997 and subsequently, since 1 January 2006, the update to the 2002 version of the ICD-9-CM classification has been adopted, in compliance with the ministerial decree of 20 November 2005. The classification used, which represents the Italian translation of the version 2007 of the American ICD-9-CM classification, was prepared by the Health section of the Ministry of Work, Health and Social Policies, and has been published by the Italy's *Istituto Poligrafico e Zecca dello Stato*[6]. It has been used uniformly throughout the italian national territory starting from the 1st of January 2009 for the codification of the diagnosis, main and secondary, and the main and secondary procedures contained in the SDOs.

The ICD-9-CM system contains two classifications, one for diseases and one for procedures, each of which is constituted by an alphabetical index and a systematic list; the following four parts are configured as follow:

- alphabetical index of diseases and traumas
- systematic list of diseases and traumas
- alphabetical index of surgical interventions and diagnostic and therapeutic procedures
- systematic list of surgical interventions and diagnostic and therapeutic procedures

In addition there are two additional classifications:

- factors influencing the state of health and the use of the facilities health (V codes)

[6]It is responsible for printing official State publications, including the Official Journal, the State Values, including stamps (through the Values Paper Workshop) and coin denomination. IPZS also operates in the field of security anti-counterfeiting (electronic identity card, electronic passport, electronic residence permit), in the printing of vehicle license plates and in Internet services, for example, by creating and managing institutional sites and databases.

- external causes of traumas and poisoning (codes E)

The alphabetical index and the systematic list of the two classifications are designed to complement each other: single clinical, pathological or procedural terms are searched in alphabetical indexes and the correctness of the codes assigned are then verified with all the accessory indications given in the relative systematic lists.

The dictionary is made by 16'212 codes and official diagnosis definition. In the example above we can see the relation between code and the associated diagnosis.

71873 - *Developmental dislocation of joint forearm*

The ICD-9-CM Dictionary translated in italian by the Ministry of Health is available at this link: `http://www.salute.gov.it/imgs/C_17_pubblicazioni_2251_ulteriorioallegati_ulteriorioallegato_6_alleg.xls`

In order to compare the domain-specific word embedding, we used two general purpose pre-trained model: a general Wikipedia italian dump and a model trained with TED corpus.

As Wikipedia is one of the first choice when searching for a large corpus of general data, we gathered word vectors trained with skipgram's word2vec by the Human Language Technologies(HLT)[7].

The latest italian Wikipedia dump can be found at this link:

- `https://dumps.wikimedia.org/itwiki/latest/itwiki-latest-pages-articles.xml.bz2`

Whereas the word vectors trained with skipgram's word2vec are available at this link (In appendix A.2.1 you can find an example of how to load such vectors):

- `http://hlt.isti.cnr.it/wordembeddings/skipgram_wiki_window10_size300_neg-samples10.tar.gz`

TED stands for "Technology, Entertainment, Design" and it is the name of conferences held worldwide with the slogan "ideas worth spreading". Video recordings of the talks at TED conferences are available on TED's website with transcriptions and translations created by volunteers. The transcripts of more than 1,900 talks are available, and they are (partially) translated into 107 languages. Transcriptions and translations are released to the public after a review process completed by volunteers, thus minimizing possible fluctuation of the data quality. The resources of TED Talks (transcripts, translations, videos, and audio) are accessible on the website for free, and they are allowed to be

[7]Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", Consiglio Nazionale delle Ricerche - Pisa, Italy.
Site link: `http://hlt.isti.cnr.it/`

redistributed under the Creative Commons "Attribution - Non Commercial - Non Derivative" license. According to this license, the materials are available for non-commercial academic and educational purposes as long as they are properly credited and the talks are kept unedited. In fact, there already exist projects aiming to use TED Talks as a corpus. The Web Inventory of Transcribed and Translated Talks (WIT3), for instance, compiled a dataset of TED Talks for conducting cross-lingual machine translation tasks (**?** **?** [**?**]). Also, the TED-LIUM project released a dataset primarily for training acoustic models (**?** **?** [**?**]). It consists of 1,495 audio talks and their transcripts accompanied by 159,848 dictionary entries to show pronunciation. Another TED-as-corpus project is the NAIST-NTT TED Talk Treebank, which developed a manually annotated treebank of 10 TED Talks (**?** **?** [**?**]).

The corpus was processed by Hermann and Blunsom at Oxford University for 'Multilingual Models for Compositional Distributed Semantics', 2014 [**?**]. They developed a massively multilingual corpus based on the TED corpus. This corpus contains English transcriptions and multilingual, sentence-aligned translations of talks from the TED conference. While the corpus was aimed at machine translation tasks, they used the keywords associated with each talk to build a subsidiary corpus for multilingual document classification as follows. At the end it amounts to 1'678'219 non-English sentences (the number of unique English sentences is smaller as many documents are translated into multiple languages and thus appear repeatedly in the corpus). Each document (talk) contains one or several keywords.

The italian TED corpus data can be found at this link:

- `https://wit3.fbk.eu/download.php?release=XML_releases/xml&type=zip&slang=ted_it-20160408.zip&tlang=undefined`

Whereas the pre-trained vectors model can be found at this link:

- `http://camoes.lx.it.pt/amartins/projects/data/multilingual_embeddings.it`

CHAPTER 7

Experiment

chapter 8

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. Google-News, 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 9

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 9.1, 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,

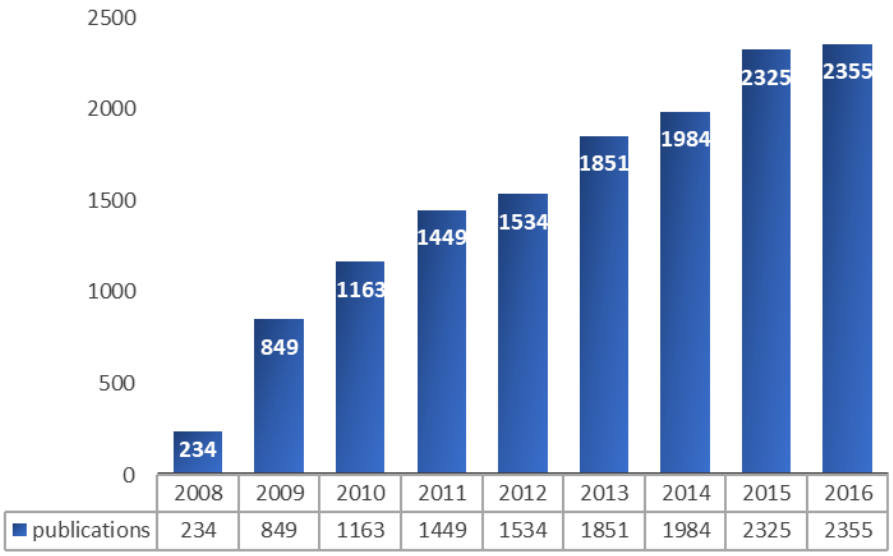

Figure 9.1: The number of publications in PubMed using the query word "*text mining*" or "*literature mining*" in the title or abstract over the last years. Search detail: *text mining* [Title/Abstract] or *literature mining* [Title/Abstract].

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

[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.

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.

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.

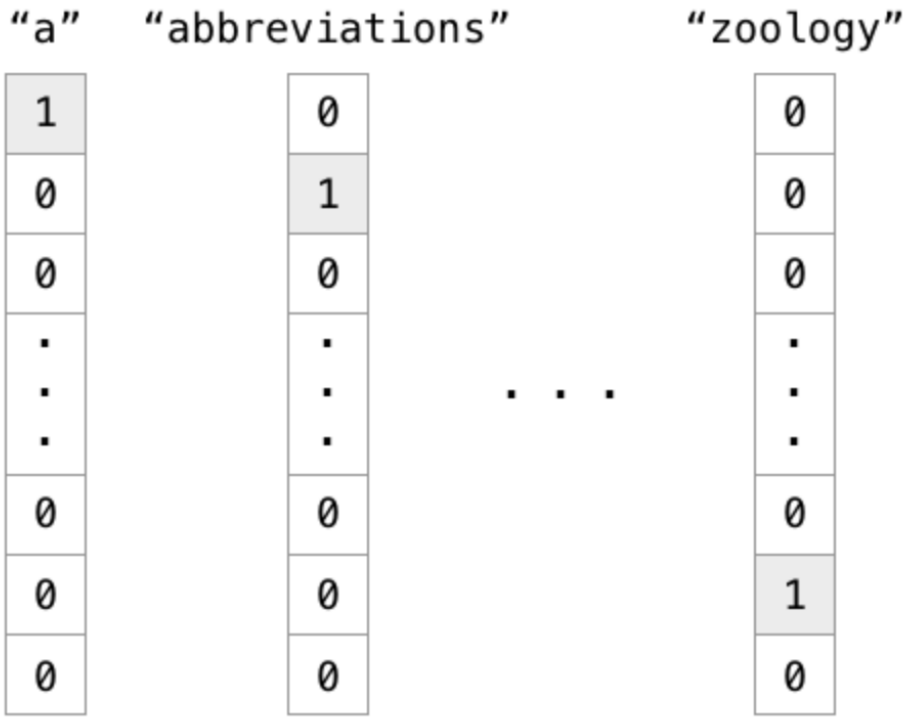

Figure 9.2: One Hot Vector: A simple and easy way to implement word vectors. 1 Bit set to 1 and all the others to 0. The dimension of the vector depends on the size of the vocabulary in input.

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

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

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.

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 $\delta(z)$ 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 9.3]

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 9.4]

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 9.5 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

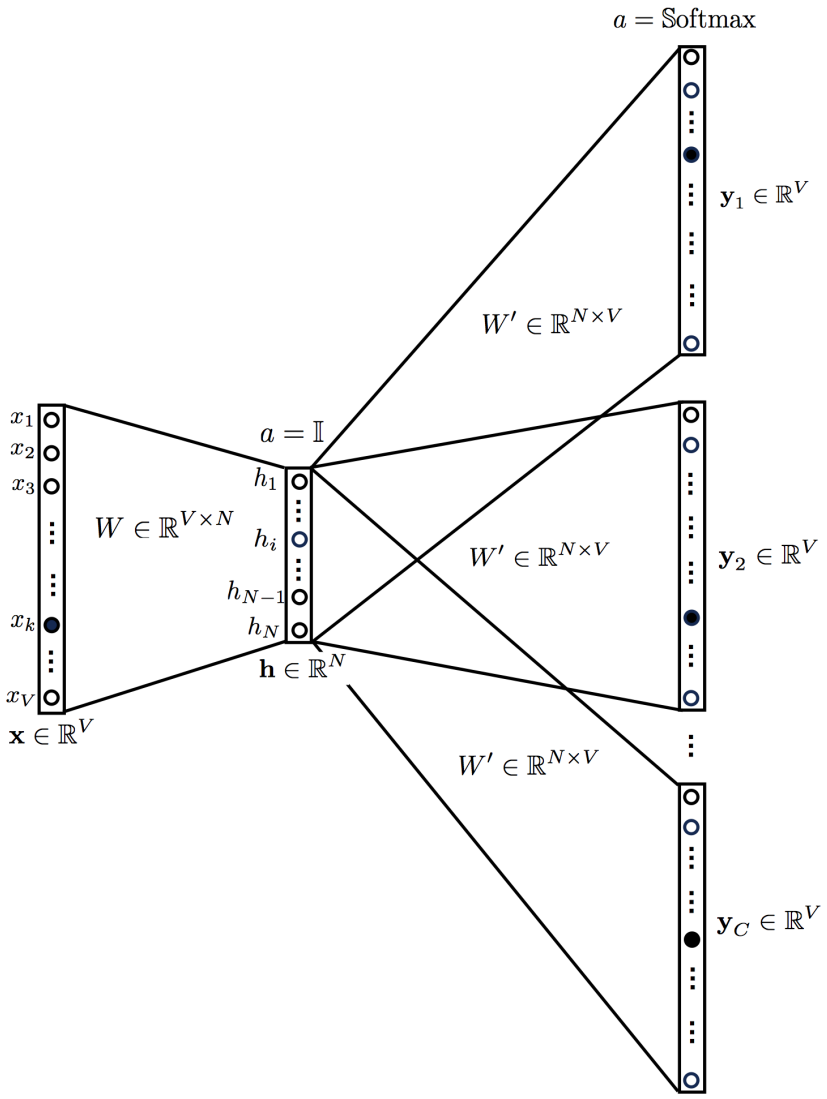

Figure 9.3: Skip-gram: The word embedding for the target words can be obtained by extracting hidden layers after feeding the one-hot representation of that word into the network.

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

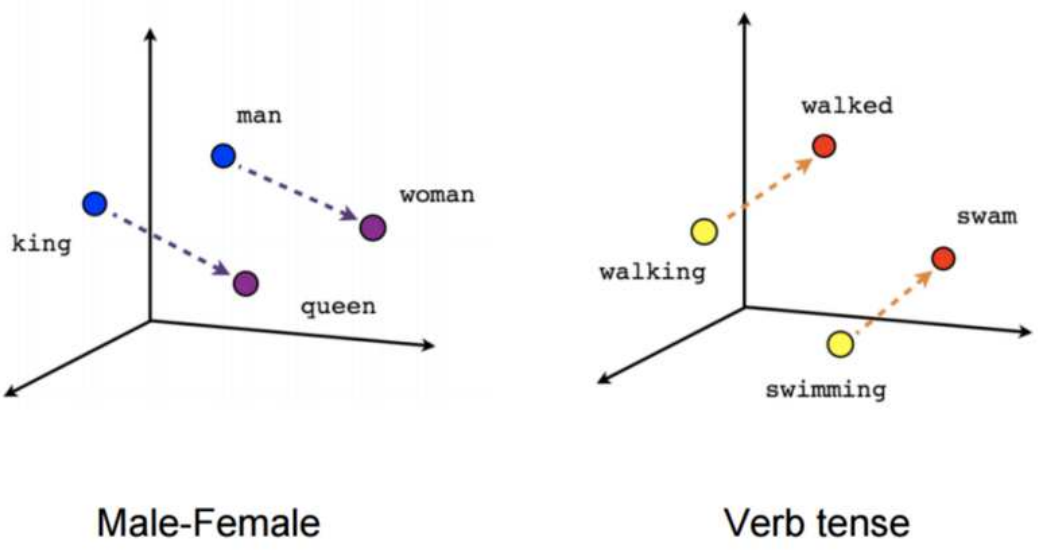

Figure 9.4: Skip-gram: the vectors are more "meaningful" in terms of describing the relationship between words.

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

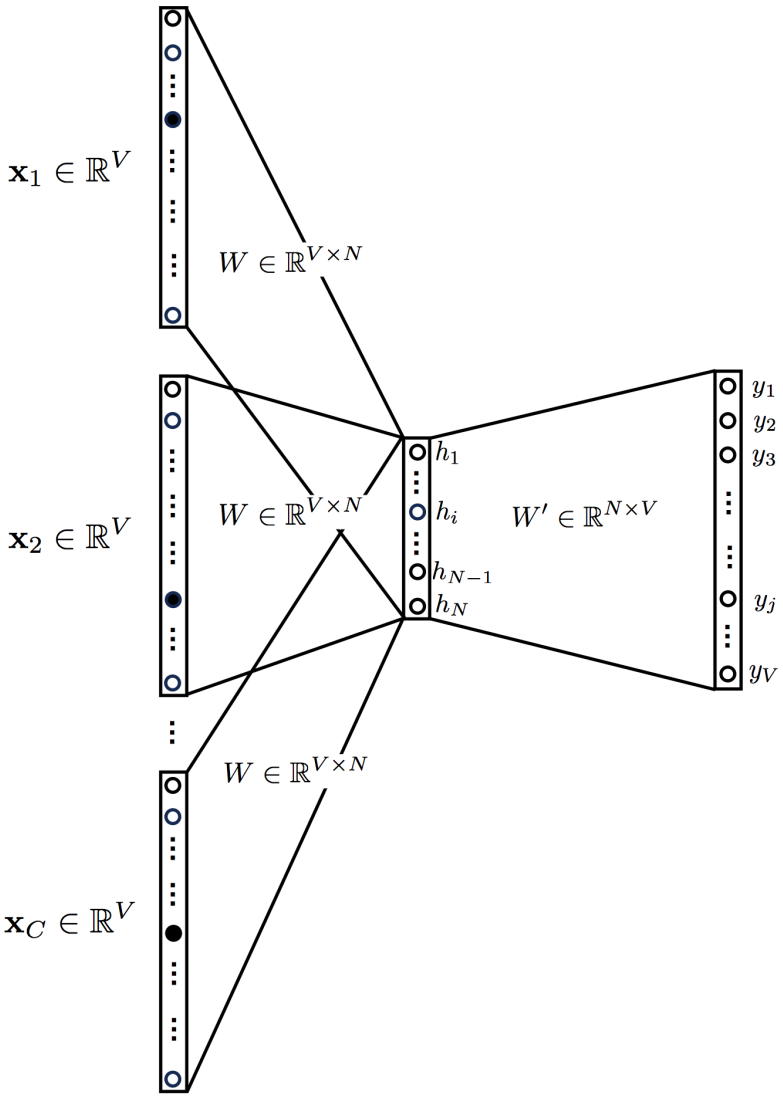

Figure 9.5: The main difference between CBOW and Skip-gram is that CBOW swaps the input and output.

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 10

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. Google-News, 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] 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.

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

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

[5] 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.

[6] 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.

[7] 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.

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 fornitaci per superare senza ostacoli 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 un flagello 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).

Example: *PetScan query* depth=*2,* category=*Salute and* language=*it*

```
https://petscan.wmflabs.org/?language=it&project=wikipedia&depth=2&categories=
Salute&combination=subset&negcats=&ns%5B0%5D=1&larger=&smaller=&minlinks=
&maxlinks=&before=&after=&max_age=&show_redirects=no&edits%5Bbots%5D=both&
edits%5Banons%5D=both&edits%5Bflagged%5D=both&page_image=any&ores_type=any&
ores_prob_from=&ores_prob_to=&ores_prediction=any&templates_yes=&templates_
any=&templates_no=&outlinks_yes=&outlinks_any=&outlinks_no=&links_to_all=
&links_to_any=&links_to_no=&sparql=&manual_list=&manual_list_wiki=&pagepile=
&wikidata_source_sites=&subpage_filter=either&common_wiki=auto&source_combination=
&wikidata_item=no&wikidata_label_language=&wikidata_prop_item_use=&wpiu=any&
sitelinks_yes=&sitelinks_any=&sitelinks_no=&min_sitelink_count=&max_sitelink_
count=&labels_yes=&cb_labels_yes_l=1&langs_labels_yes=&labels_any=&cb_labels_
any_l=1&langs_labels_any=&labels_no=&cb_labels_no_l=1&langs_labels_no=&format=
json&output_compatability=catscan&sortby=none&sortorder=ascending&regexp_
filter=&min_redlink_count=1&doit=Do%20it%21&interface_language=en&active_
tab=tab_output
```

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.

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.

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.

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()
```

[1] https://matplotlib.org/api/pyplot_api.html
[2] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

List of Figures

List of Tables