POLITECNICO DI TORINO

Master Degree Course in Computer Engineering

Master Degree Thesis

Semantic Analysis to Compute Personality Traits from Social Media Posts

Supervisor:
Prof. Maurizio Morisio

Candidate
Giulio Carducci
student id: 225395

Internship Tutor
Dott. Ing. Giuseppe Rizzo

Academic Year 2017-2018

Contents

Chapter 1

Introduction

We are what we do, like, and say. Personality is what makes a person different from another and affects the way we think, speak, and behave. For years, psychologists and researchers have focused their studies with the intention to derive a universally accepted model that describes the personality of an individual, which is nowadays known as the Five Factor Model. The study of human personality has a long and rich history, whose first traces date back to the ancient Greece. Personality influences academic and job performance, social and political attitudes, the quality and stability of social relationships, physical health and mortality, and risk of mental disorder. This is the reason so much efforts have been put on studying this field. Knowing the personality of an individual, and more specifically the correlation between personality and behavior, opens the way to a huge number of applications, and in the Age of Information, in which data is the protagonist, this implication is of particular importance.

During the last years, Social Networks have seen an enormous growth in terms of number of active users and their influence on people and ordinary life. In just about fifteen years, Facebook passed from one million monthly active users to over two billions, with an outstanding growth of almost 200000%[1]. A similar trend, yet with lower numbers, has been registered for Twitter, which passed from 30 millions monthly users in 2010 to 330 millions in 2017[2], with a growth of 1000%, and about 6000 tweets tweeted every second, for a total of roughly 500 millions tweets a day. Analogous numbers can be seen for other social networks such as Instagram, Snapchat, Tumblr and more.

[1] https://newsroom.fb.com/

[2] https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

Those online platforms have become a common place to share personal opinions or knowledge, stay in contact with other people, read news or articles, or simply spend some free time, and although they were originally intended as a pastime activity, their huge success sometimes changed the primary reason of their use. Important people (also called "influencers") such as singers, actors, or even politicians, regularly use social networks to increase their follow and advertise their public figure, and sometimes use their influence to manipulate the masses, hence the term influencers. Companies make use of those platforms to advertise their brands and carry out marketing campaigns, which in some cases are more effective than those performed on more traditional medias such as television and radio, because they are able to exploit more information (e.g. demographic, social network usage) about their audience, allowing to target a particular group of users in a more effective manner with respect to conventional advertisement campaigns. The appearance and content of users' personal profiles are often analyzed by possible recruiters to outline the main characteristics of a person, since personal pages often reflects the personality (here intended with a more general allusion) of the owners.

The importance of human personality and the widespread of social networks have motivated the work of this Thesis, which we are going to introduce in the following chapters. We aim to predict the personality of Twitter users using only the information that can be inferred from textual data, applying semantic analysis to the raw text of their tweets. More specifically, we use distributed representations of words, known as word embeddings, to transform tweets into vector of real numbers, which are then used as input features to a supervised learning approach in order to derive a predictive model that is capable of inferring the personality of unknown users. This work is based on the well known assumption that the personality characteristics of an individual will come to be encoded in terms of the particular language features that he or she tends to adopt. That is, the words and expressions we use are representative of ourselves.

Personality prediction is a task that can be accomplished with many different types of input data, for example page likes [1], information and interests displayed in the personal profile page [49], or even the choice of profile picture [2]. However, due to privacy settings of the different social networks, in most, if not all the cases, it is required to ask for the user permission before being able to automatically collect such pieces of information, and this may be unfeasible at a large scale. Instead, by relying only on publicly available data, namely tweets from Twitter, it is possible to avoid this problem and efficiently apply such analyses at large scale. The choice of Twitter as the source social network hence comes from this requirement; we wanted to work only with data that is public and can be seen by anyone on the social network.

By using social networks, people share a wide range of information with their audience, which can be used for a large number of applications. There are many

studies in literature that exploit such data to perform user profiling, personalized marketing, recommendations, or simply statistical analyses. Results achieved on those studies are often outstanding and highly promising, and demonstrate that such analyses can be very effective and accurate, sometimes even more than humans [28].
However, while on the one hand the availability of social network data is clearly an advantage for some goals, on the other hand it poses some serious privacy issues related to the fact that users' data can be freely seen, downloaded and analyzed, and scientific research, like the case of this thesis or the above mentioned examples, is just one of all the possible uses of those pieces of information.
Indeed, such data can be acquired and processed with malicious intent. Most of the times this is done for addressing the user with targeted spam messages, but it could be also done for more serious attacks such as phishing and social engineering, with the final objective of fraud and robbery. Those malevolent activities could also be done without the need of exploiting social network data, but this helps making the offenses more threatening and likely to succeed.

User data may also be used to infer some personal information or characteristics that people may want to hide. For example, even when there is no direct association between homosexuality or pregnancy and social network data, it may be still possible to guess those two aspects with a certain degree of accuracy [1], and in some cases this may harm the privacy of the individual. This has to be taken into account even when the person has expressed the explicit consent to use its data. For the maximum transparency, one should always inform users about which data he is going to use and for what purpose. Moreover, privacy settings of the websites may deny other people to see certain information, but they do not prevent the website itself, and sometimes third party apps, to access them. Stutzman et al. [3] analyzed the behavior of a subset of users during the early years of Facebook, and discovered that while there has been a common trend that saw users seek an increasing privacy aware behavior the over the years, changes made by Facebook on its platform resulted instead in a general increase of disclosures of personal information of various type. For this reason, the actual audience of someone's disclosure is in fact different, and larger, than the expected one.
Inferring user information from publicly available data may be even more dangerous since the user is unaware that his data is being used for analyses (yet one could argue that he chose to share such data, accepting the fact that others would be able to see it). For example, the website *pleaserobme.com* combines location data from Twitter and Foursquare, which is intended to be shared only with a small circle of contacts, to check whether someone is at home or not, and presenting this information to the whole internet community, which may include burglars. The website was created for raising awareness about over-sharing and its authors did not have malicious intent, but someone else may think it differently.

For what concerns our work, we are only interested in the scientific aspect of the analysis, namely, whether it is possible to accurately predict someone's personality by only using the language features presented in a social network context. The data we use to train the predictive model is anonymized, and we test the approach on a group of participants who gave their explicit consent on downloading their data, knowing the type of analysis that we were going to carry out.

The remainder of this document is organized as follows: in Chapter 2 we present the related work and state of the art in literature, in Chapter 3 we describe the model of personality that we adopt, in Chapter 4 we present the type of semantic analysis that we apply to the text, while in Chapter 5 we show how our application works, and how it is possible to derive a score from a tweet. In Chapter 6 we report the results obtained with different approaches and the result of testing the application on a real sample of Twitter users. Finally, in Chapter 7 we discuss the possible improvements of our work and its implications.

Chapter 2

Related Work

Numerous studies have demonstrated the correlation between personality traits and many types of behavior. This includes job performance [4], psychological disorders [5, 6], and even romantic success [7]. Significant correlation were also found between personality and preferences. Studies showing connections between personality and music taste are well established in literature [8, 9, 10, 11], while Jost et al. [12] found that knowing the personality traits of an individual allowed to predict whether they would be more likely to vote for McCain or Obama in the presidential election of 2008 in the United States. Cantador et al. [13] presented a preliminary study in which they analyzed preferences of roughly 50,000 Facebook users who expressed their interests about sixteen genres in each of these domains: movies, TV shows, music and books. Such results can be very valuable to enhance personalization services in several domains.

Profiling a person by using words she uses has been extensively studied in language psychology. In fact, a central assumption in language psychology is that the words people use reflect who they are. For instance, in [14, 15] authors investigated and observed that the ways people spoke were related to their physical and mental health problems. With the increasing computer advances, efforts were made in attempting to capture psychological themes or people's underlying emotional states that might be reflected in the words they used (e.g., [16]). Other psychological and linguistic studies have supported the intuition that the personality traits of individuals are implicitly encoded in the words used to shape a sentence [17, 18].

The volume of research on computational personality recognition has grown steadily over the last years, there have also been dedicated workshops [19, 20]. The widespread of social media platforms has inspired researchers to move towards these platforms to seek useful information to be used for personality prediction. Many studies showed correlation between personality and online behavior

[21, 22, 23], and there is an excellent corpus in literature about inferring personality traits from social networks. Quercia et al. [24] were the first who explored at large the relationship between personality and use of Twitter, they also proposed a model to infer users' personality based on just following, followers, and listed count numbers. Similarly, Jusupova et al. [25] used demographic and social activity information to predict personality of Portuguese users, whereas Liu et al. [26] proposed a deep-learning based approach to build hierarchical word and sentence representations that is able to infer personality of users from three languages: English, Italian, and Spanish. Van de Ven et al. [27] based their analyses on LinkedIn, a job-related social networking site, in order to verify whether the predictive power of the information it contains is comparable to the one of more traditional social networks such as Facebook or Twitter. However, they did not find strong correlations between personality traits and user profiles, except for Extraversion. YouYou et al. [28] demonstrated that computer-based judgments about an individual can be more accurate than those made than friends, spouse, and even the individual himself, if sufficient data is available. The collected human personality judgments from the participant's Facebook friends using a short personality questionnaire, and evaluating their accuracy using three criteria: self-other agreement, inter-judge agreement and external validity.

First approaches on personality prediction were mainly based on SVM, using syntactic and lexical features [18, 29]. It was just in the last years that researchers moved towards deep learning. Kalghatgi et al. [30] and Su et al. [31] employed neural networks by feeding them a number of meticulously hand-crafted features, the first about syntax and social behavior while the latter regarding grammar and LIWC annotations extracted from a dialogue. About different approaches, neural networks could be employed to perform semantic analysis by automatically deriving a distributed representation of a sentence starting from raw text. For example, Liu et al. [26] used two recurrent neural networks (RNNs) to automatically obtain first word representation from characters, and then sentence representation from words, extending the C2W model that was originally proposed by Ling et al. [32]. Majumder et al. [34] used a CNN to derive a fixed-length feature vector starting from word2vec word embeddings [35], which they extended with 84 additional features from Mairesse's library [36]. For classification, the so computed document vectors are fed to a multi-layer perceptron (MLP) and in a different study case to a polynomial SVM classifier.
In contrast to the above approaches, we adopt a dictionary of existing word representations generated by the skipgram model implemented in FastText. We then combine word representations using geometric manipulations of the vectors, ultimately we feed them into a radial SVM classifier.

Distributed representations of words are capable of successfully capturing meaningful syntactic and semantic properties of the language and it has been shown [100]

that using word embeddings as features could improve many NLP tasks, such as information retrieval [37, 38], part-of-speech tagging [39] or named entity recognition (NER) [40]; Kuang and Davidson [41] learned specific word embeddings from Twitter for classifying healthcare-related tweets, while Yang et al. [42] use word embeddings in a convolution neural network for Twitter election classification. Since learning those word representations is a slow and non-trivial task, already trained models can be found in literature; state-of-the-art embeddings are mainly based on deep-learning [35, 43], but other techniques have been previously explored, for instance spectral methods [44, 45]. If correctly trained, word embeddings could also improve performances on sentiment analysis. In fact, existing models typically ignore the sentiment information; words like *good* and *bad* are mapped into close vectors, due to their similar usages and grammatical roles. Tang et al. [46] overcome this issue by learning sentiment specific word embeddings. They extend the model from Collobert et al. [47] by incorporating sentiment information in the neural network.
In the attempt to shorten the gap between two languages, Zou et al. [48] proposed a method for learning bilingual word embeddings for Chinese and English, using word alignments to maintain the translational equivalence between embeddings of similar word in the two languages.

There is a wide number of studies that does not use lexical features but it is rather based on social media platform metadata. Kosinski et al. [1] applied singular value decomposition (SVD) on a user-like matrix containing associations between users and Facebook pages, then predicted a wide range of personal attributes implementing a regression model with the top 100 SVD components; Golbeck et al. [49] collected everything available from users' Facebook profile and activities, which they extended with a number of composite features; Quercia et al. [24] based their analyses only on three numbers publicly available on Twitter: following, followers and listed count. Based on these, users are classified as popular, listeners, highly-read, and two types of influencers, each of these groups having different personality characteristics.

Social media platforms are rich sources of information not only for personality prediction. Sentiment analysis has attracted increasing interest in recent years [50, 51, 52, 53]. It consists of classifying the sentiment polarity of a sentence, as positive, negative or neutral. Dai et al. [54] explored numerous feature engineering techniques for detecting adverse drug reactions (ADR) from Twitter posts. Studies on demographics are also worth of note, in particular for those websites where demographic information is mostly unavailable (e.g. Twitter). Chamberlain et al. [55] and Zhang et al. [56] successfully inferred users' age based on their interaction on the social media platform. The first extracted ground truth labels from users' descriptions and generalized it to the entire Twitter network of about 700 million users using a probabilistic model for age inference based on the profiles they follow,

while the latter used semantic analysis of tweets to which they incorporated online interaction information to classify users into 5 distinct age groups.
Other studies on enhancing social data with demographic attributes include gender [57], location [58], ethnicity [59], and political affiliation [60].

Chapter 3

Five Factor Model

The Five Factor Model (FFM), also called the Big Five, is the most widely accepted model of personality. It integrates a wide array of personality constructs in an efficient and comprehensive manner [61]. For decades, several independent groups of researchers put their effort on deriving a general and comprehensive personality construct, and they all achieved similar results, suggesting that the hypothesis of the existence of an underlying model describing human personality was well founded.
The Five Factor Model defines five traits that are general enough to model the personality of an individual at a high level: *Openness*, *Conscientiousness*, *Extraversion*, *Agreeableness*, and *Neuroticism*.

The FFM provides a common language for psychologists from different traditions, a basic phenomenon for personality theorists to explain, a natural framework for organizing research, and a guide to the comprehensive assessment of individuals that is of value to educational, industrial/organizational, and clinical psychologists [61].

Factor analysis is a statistical method that is used to describe variability among observed variables in terms of a potentially lower number of unobserved variables, which are denoted as *factors*. The purpose of factor analysis is to search for those factors and model the observed data as a linear combination of unobserved latent variables. This allows, for example, to reduce the size of the variables in a dataset and to create simpler models without losing information. As researchers applied factor analysis to personality survey data they found a number of recurrent factors; this motivated further studies towards a general personality model, until the FFM was advanced.

The theory behind the FFM is based on the association between words rather than neuropsychological experiments, using descriptors of common language. In

fact, word choice and language features typical of an individual reflect his/her personality. This is what states the *lexical hypothesis*, introduced below.

Lexical Hypothesis The lexical hypothesis is a thesis mainly adopted in the field of personality psychology, that is, a branch of psychology that focuses on personality and its manifestation and variation among individuals. It is defined by two postulates:

1. Personality characteristics that are important to a group of people will eventually become part of that group's language.

2. The main personality characteristics of an individual are more likely to be encoded in the language as a single word [62].

Its origins date back to the late 19th century, with Sir. Francis Galton being one of the first scientists who acknowledged the hypothesis and based their research on it [65]. It then received a higher attention during the 20th century [63]. Lexical hypothesis is a major foundation of the Five Factor Model and has been the base for many studies on personality traits in numerous languages and contexts [64].

3.1 The Model

The five factors defined by the FFM are:

- Openness to Experience
- Conscientiousness
- Extraversion
- Agreeableness
- Neuroticism

They are often referred to with the acronym OCEAN. Figure 3.2 represents the main characteristics of the traits and properties related to low and high scores. A personality trait is assigned a numeric score in a range of values, for example 0-1 or 0-5. The scores of all five traits may be graphically represented as a pentagonal radar chart, as in Figure 3.1.

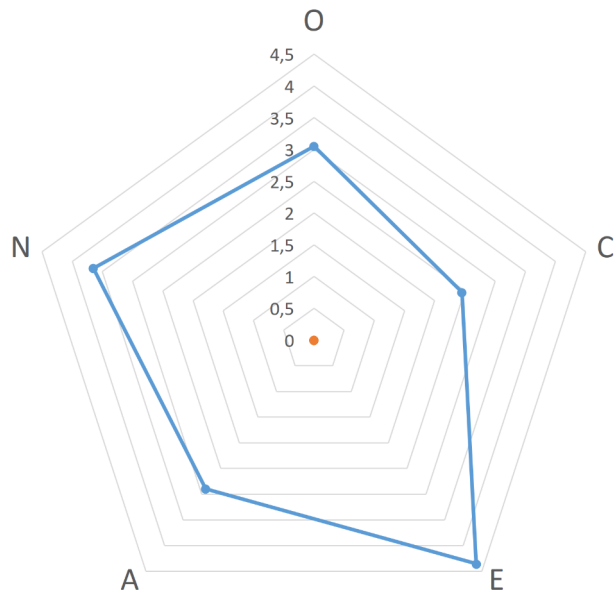

Figure 3.1. Graphical representation of the personality sphere of an individual, where scores are expressed in the range 0-5.

Openness To Experience

It describes an individual's willingness to try new things and experiences. Open people are intellectually curious and tend to feel a higher appreciation for art and beauty; they are also adventurous and creative. They have a wide variety of interests and tend to be more imaginative. People low in Openness prefer routine over variety and tend to dislike new ideas and experiences. They are much more traditional and may struggle with abstract thinking.

Conscientiousness

It involves the way we control, direct and regulate our impulses. Those high in C tend to be organized, methodic, reliable, avoid troubles and achieve success through accurate planning, though they might appear as compulsive perfectionists. Individuals low in C are less organized, more likely to procrastinate, impulsive, and tend to avoid sticking to rules and schedules. High Conscientiousness is also related to success in school and in career, as well as leadership.

Extraversion

It describes how an individual interact with others, or more generally with the external world. Extroverts people seek and enjoy company of others, tend to be enthusiastic and action-oriented. They are talkative, assertive and openly express their emotions. Those low in Extraversion are quiet, introspective, and prone to contemplation rather than action. They prefer being alone and find it difficult to start conversations and be in social situations. Their lack of social involvement should not be interpreted as shyness or depression; the introvert simply needs less stimulation than an extrovert.

Agreeableness

Describes how well people get along with others. While Extraversion concerns sources of energy and the pursuit of interactions with others, Agreeableness concerns the orientation to others. Those high in A tend to be friendly, cooperative and passionate, and have an optimistic view oh human nature. Those associated with low A have little interests in others' feelings and problems, tend to be suspicious, unfriendly and uncooperative. They are also likely to be sarcastic and offensive. Although not all people low in Agreeableness are cruel or unfriendly, they are not likely to leave others with a good feeling.

Neuroticism

It represents individual differences in the tendency to experience distress, and in the cognitive and behavioral styles that follow from this tendency. Those high in Neuroticism experience chronic negative effects [66] and are prone to the developments of numerous psychiatric disorders [67]. High N scores correlate with low self-esteem, anxiety, worry, and people tend to get easily angered and be unsure of themselves. Individuals low on Neuroticism are more likely to feel sure of themselves, confident, and adventurous. They are also calm, even-tempered and relaxed.

3.1.1 Personality Facets

The Five Factor Model is only a high level representation of human personality [61, 92], and provides a good description with as low as five dimensions. However, it is possible to provide a more accurate and exhausting characterization. There is in fact, for each personality trait, a number of underlying and lower level dimensions that are capable of capturing more subtle nuances far more effectively than the FFM [66, 67]. We refer to these dimensions as *facets*.
Costa and McCrae, in their Revised NEO Personality Inventory (NEO-PI-R, [68])

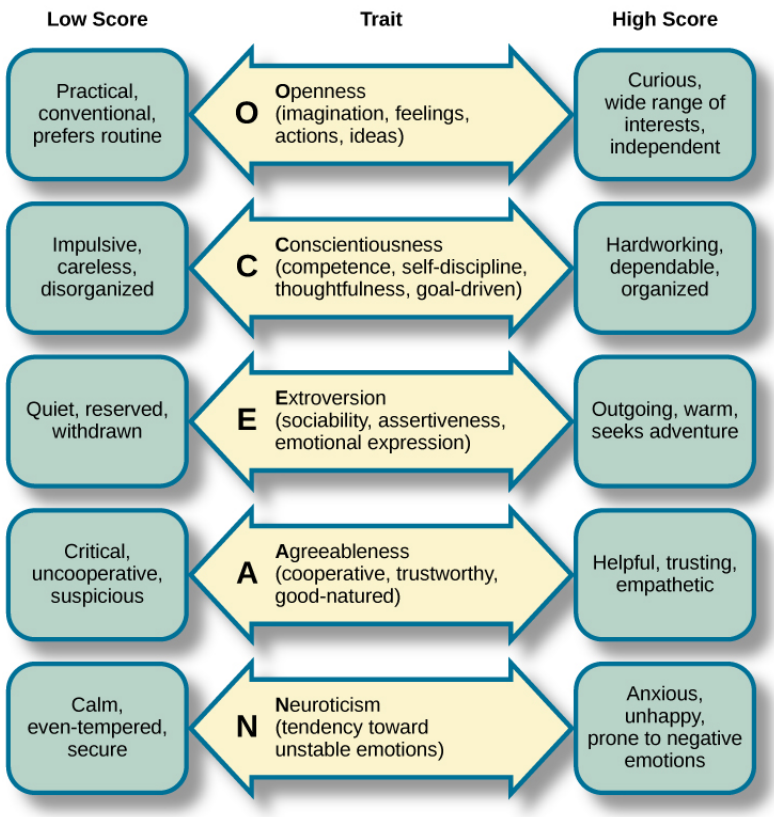

Figure 3.2. OCEAN traits characteristics

defined a total of 30 facets, 6 for each personality factor; they are reported in Table 3.1.

Personality facets are way more effective at describing the personality sphere of an individual with respect to just the scores of the Five Factor Model, which only describes individuals at the highest hierarchical level. By going down the hierarchy, and hence refining the classification by moving it at a more granular level, it is possible to describe a wider range of human aspects. For example, Extraversion measure how a person poses towards other people and the external world in general. All those that are high in extraversion share similar characteristics and general behaviors, but they may differ in warmth or excitement-seeking, in gregariousness or assertiveness. Every person is extroverted in his own way, yet they are all extroverts.

Factor	Underlying Facets	
O	Imagination	Willingness to Experiment
	Artistic Interests	Intellectual Curiosity
	Depth of Emotions	Tolerance for Diversity
C	Sense of Competence	Achievement Striving
	Orderliness	Self-Discipline
	Sense of Responsibility	Deliberateness
E	Warmth	Activity Level
	Gregariousness	Excitement-Seeking
	Assertiveness	Positive Emotions
A	Trust in others	Compliance
	Sincerity	Modesty
	Altruism	Sympathy
N	Anxiety	Self-Consciousness
	Angry Hostility	Self-Indulgence
	Moodiness/Contentment	Sensitivity to Stress

Table 3.1. Underlying personality facets for each factor of the FFM. Defining more "classes" for each trait allows to describe individuals with a higher level of accuracy.

3.2 Measuring Personality Scores

Personality is usually assessed with ad-hoc tests, also called questionnaires, which are based on scrupulously designed scales. A personality questionnaire features a variable number of questions (or items), which depends on the questionnaire itself, that are used to measure the personality of the individual answering them. Each question describes common life situations and behaviors, and the individual should indicate the extent to which statements describe himself using a Likert scale [69]. Questions are positively or negatively related to a specific trait or facet. For example, the question "I dislike being the center of attention." is clearly related to Extraversion, while the question "I get angry easily." is related to Neuroticism. Then, a specific value is assigned to each answer, based on the score indicated on the Likert scale. Finally, the score for each personality trait is computed by aggregating the values of all the answers related to that specific trait. An example of questions extracted from a questionnaire is showed in Figure 3.3.

Psychologists and researchers have developed a large number of questionnaires and scales during the years. For example, the NEO Five Factor Inventory from McCrae and Costa (NEO-FFI, [68]) is a 60-item questionnaire that is able to assess personality with as low as 12 items per trait, although it does not measure facets. For a more accurate test, the researchers proposed the Revised NEO Personality Inventory (NEO-PI-R, [68]), which counts 240 items, 48 for each trait, and is also able to define facets. Those tests administer a fixed number of questions (60 and 240) to which the participant should answer on a scale that goes from "strongly disagree" to "strongly agree".
A rich source of questionnaire items and rating scales is available on the International Personality Item Pool[1] (IPIP, [70]). The site includes over 3000 items and 250 scales that researchers can consult and implement for free in their own personality test. The website also includes two online tests, one featuring 120 items and the other (IPIP-NEO) 300.

As one can expect, the accuracy of a test rises accordingly to the number of questions that it administers. Questionnaires results, however, may not be fully accurate. This is because they only rely on self-ratings, that is, the perception of the individual's own personality. In order to get more precise and meaningful scores, one should also consider peer ratings (e.g. from a friend) and observer ratings (e.g. from a layperson—an external observer).

3.3 History

There are two main approaches in the history of personality studies. These are based on different hypothesis and derivation methods, but they both lead to similar models. The *lexical tradition* is based on the lexical hypothesis, and personality traits are derived using factor analysis over a lexicon of trait-descriptive terms, whereas *personality questionnaires* are based on accurately designed scales. At some point in the history of research, the two approaches merged, leading to the present FFM. We briefly review the history and milestones of both approaches.

3.3.1 The Lexical Tradition

As we said, Sir. Francis Galton was among the first scientists who based their research on the lexical hypothesis [65]. He created a lexicon of personality-related words and conducted a preliminary analysis on the extent to which trait terms share their meanings. Galton's lexicon was later narrowed down first by Allport and Odbert [71] and then by Norman [72]. Relationship among personality terms

[1] http://ipip.ori.org/

1.	Worry about things.	Very Inaccurate	Moderately Inaccurate	Neither Accurate Nor Inaccurate	Moderately Accurate	Very Accurate
2.	Make friends easily.	Very Inaccurate	Moderately Inaccurate	Neither Accurate Nor Inaccurate	Moderately Accurate	Very Accurate
3.	Have a vivid imagination.	Very Inaccurate	Moderately Inaccurate	Neither Accurate Nor Inaccurate	Moderately Accurate	Very Accurate
4.	Trust others.	Very Inaccurate	Moderately Inaccurate	Neither Accurate Nor Inaccurate	Moderately Accurate	Very Accurate
5.	Complete tasks successfully.	Very Inaccurate	Moderately Inaccurate	Neither Accurate Nor Inaccurate	Moderately Accurate	Very Accurate
6.	Get angry easily.	Very Inaccurate	Moderately Inaccurate	Neither Accurate Nor Inaccurate	Moderately Accurate	Very Accurate
7.	Love large parties.	Very Inaccurate	Moderately Inaccurate	Neither Accurate Nor Inaccurate	Moderately Accurate	Very Accurate
8.	Believe in the importance of art.	Very Inaccurate	Moderately Inaccurate	Neither Accurate Nor Inaccurate	Moderately Accurate	Very Accurate
9.	Would never cheat on my taxes.	Very Inaccurate	Moderately Inaccurate	Neither Accurate Nor Inaccurate	Moderately Accurate	Very Accurate
10.	Like order.	Very Inaccurate	Moderately Inaccurate	Neither Accurate Nor Inaccurate	Moderately Accurate	Very Accurate

Figure 3.3. First ten questions of the complete NEO-IPIP personality inventory. Each question is related to a specific trait, and the user can pick the answer that best describes himself.

initially proposed by Galton were further investigated by subsequent researchers aiming to discover the nature of those relations and construct a structural representation of personality descriptors. Among the first of them was L.L. Thurstone, who was a pioneer in factor analysis and whose initial findings almost mirror the present model. In fact, he analyzed the use of sixty adjectives over 1300 raters, and concluded that five factors were sufficient to account for all sixty of the adjectives. However, he did not pursue this path afterwards. Cattell [73] began his analysis from the dictionary of trait-descriptive terms outlined by Allport and Odbert [71], from which he identified 12 factors. However, later analyses of Cattell's variables showed that only five factors were replicable [75, 74, 72, 76, 77]. In particular, Fiske [74] analyzed a set of 22 variables developed by Cattell and found five factors that replicated over self ratings, observer ratings and peer ratings. The honour of laying the foundation for the Five Factor Model is generally credited to Tupes and Christal [77], who found five recurrent factors in eight different samples:

> "In many ways it seems remarkable that such stability should be found in an area that to date has granted anything but consistent results. Undoubtedly the consistency has always been there, but it has been hidden by inconsistency of factorial techniques and philosophies, the lack of replication using identical variables, and disagreement among analysts as to factor titles." (Tupes and Christal, 1961, p. 12)

Nevertheless, despite their promising findings, the importance of these factors remained mostly hidden in the 1960s and 1970s. It was only in the 1980s that researchers from many different traditions came to the conclusion that these factors constituted the fundamental dimensions of human personality, and they could be found in self-reports and ratings, in natural language and questionnaires, in individuals of different age, gender, and country of origin [78]. The renewed interest in the topic grew rapidly and at present we can consider the theory well-established, with years of studies of supporting evidence.

3.3.2 Personality Questionnaires

It is well known that the Five Factor Model has its origins in studies of personality-related terms from natural language, as we briefly explain in Section 3.3.1. However, the lexical tradition has a very small role in the history of personality research. In fact, most personality assessment has been based on questionnaires with meticulously designed scales, which depended on the practical applications [79]. A number of instruments originated from the studies by Jung [80], Murray [81] and Sullivan [82]; individual researchers also created scales to measure different constructs they considered important (e.g. Tellegen & Waller [83]).

Many questionnaires scales were focused on measuring two main personality factors, which can be roughly associated to the present Extraversion and Neuroticism. It was clear however that these two dimensions did not exhaust the full range of personality characteristics. Costa and McCrae [84] proposed an additional dimension, the modern Openness to Experience. It was at this point that the lexical and questionnaire tradition merged, leading to the contemporary Five Factor Model [85, 86, 89]. In 1980, both Costa and McCrae and Tellegen advanced a fourth personality dimension, recognizable as Conscientiousness. The last trait, Agreeableness, might have been discovered by Leary [87].

3.4 Evidence of Comprehensiveness

At first, correspondence between lexical factors and personality traits measured by questionnaires was not clear, and the studies on support of this hypothesis were limited both in number and in the approach they used. It was in the 1980s that more accurate studies on this topic emerged. Amelang and Borkenau [88] found five similar factors in self-reports of both trait-terms and questionnaires scales for the German language; McCrae and Costa [89, 90] showed convergence for all the factors across both observers and instruments in the adult sample they examined. Similar findings have been achieved by other researchers, proving that the correspondences between personality factors in the two traditions are empirically justified.

3.5 Critiques to the Five Factor Model

Despite being the most widely accepted and adopted model of personality, and all the evidence in support of it, the FFM is not entirely critique-free. In fact, many questions have been asked during the years regarding some of its aspects. We will address the most important ones.

Too few factors

Many writers argued that five factors are not sufficient for summarizing the entire personality aspect of an individual. For example, results provided by the 16PF can lead to more accurate predictions [66]. This in not wrong. In fact, as stated in [61] and [92], the five factors of the model do not exhaust the description of personality, they only represent the highest hierarchy of trait description, providing a complete characterization only at a global level. Depending on the intended use, one may need to make further distinctions using representations at a more granular level.

Another question that theorists asked themselves is whether there are additional factors that have not been included in the model. This could be possible yet very unlikely, given the wealth of data and research supporting the comprehensiveness of the FFM.

It should be noted, however, that some psychologists (e.g. [93]) include intelligence in the description of personality. If so, it should be considered as a distinct factor.

Too many factors

Some researchers claim that less than five factors are enough to account for the whole variance in personality. Many theories have been proposed, each of them with its own characteristics, but all the proposals are mutually inconsistent. Furthermore, empirical analyses demonstrated that all five factors are needed. McCrae and Costa [90] extracted factors from 80 adjective pairs in one sample of self.reports and one of peer ratings. When fewer – or more – than five factors were extracted, they could not be matched across the two samples, whereas the match was almost perfect using five factors. Similar results have been reported by other researchers, suggesting that five is just the correct number.

Ratings versus self-reports

As pointed out by Hogan [91], there is a difference between observer rating of personality and self-reports. The first represents the individual in a public and social context, while the second reflects inner drives and dispositions. Hogan states that the FFM is only suitable for describing the former. According to McCrae and

John [61], this is a possibility that merits further research, but does not pose any direct challenge to the cross-observer invariance of the FFM.

Cognitive artifacts versus realistic description

Ratings of strangers and observations of the similarities of traits between the Five Factor Model and people's implicit personality theories induced some researchers to move this critique. They claim that the FFM could be simply a projection of researchers' cognitive biases onto the individuals they rate. However, some studies have been proposed to test this hypothesis, and despite there are still some supporters of this hypothesis, it has been rejected by the majority of researchers.

3.6 Applications of the Five Factor Model

Personality highly influences our behavior. Whatever we do, like, and say, it is because of our particular personality characteristics. For this reason, personality studies can be applied nearly everywhere, and many fields and applications may find direct use of it.
Personality characteristics of an individual are likely to indicate whether he is team working, cooperative, or open-minded, hence indicating one's suitability for certain jobs. Recruiters could better match candidates with jobs based on these aspects. Numerous studies demonstrate the existing correlations between personality and job performance [4, 94, 95].
Marketing messages could be tailored to users' personalities, creating specific advertisements that have a higher chance of being noticed. Cantador et al. [13] presented a preliminary study on about 50.000 Facebook users in which they extensively analyzed how personality correlates with preferences in different genres of the following domains: movies, books, music, and tv shows. Such findings can be very valuable to enhance personalized marketing offers and services. For this purpose, it is also possible to exploit reviews on common marketing websites, and suggest items that have been positively reviewed by users with similar personalities. Those studies are of particular interest in the field of Recommender Systems.
Personality also affects design and visual preferences, so that a website could change its layout and graphical style according to the personality of the user that is browsing it, in order to give him a better impression [96].

In completely different contexts, personality also correlates with psychological disorders. Individuals with specific trait scores may be more likely to be diagnosed with certain diseases.
In forensics, knowing personality helps reducing the circle of suspects.
In education, personality affects the way we learn, so that teachers, professors or more general teaching systems such as e-learning platforms could account for it

when presenting their contents to the students.
When processing a user search query, web engines could exploit personality information to assign a higher importance to results that contain particular keywords or come from certain websites. This could be generalized to other information retrieval and information filtering applications.

However, all the possible applications reported above need to exploit a baseline study that analyzes the correlations between personality and the particular field of application. Those are usually done by collecting a considerable sample of individuals from that field and carefully inspecting how the different personality traits of the people reflects on their different behaviors and preferences. Personality is generally assessed by experts or by asking the participants to answer a questionnaire.

Chapter 4

Word Embeddings

Natural Language Processing (NLP) is a field of computer science interested in analyzing natural language, that is, the normal and most basic form of human communication, that we use everyday when having conversation with other people or when writing articles, comments, or messages. First traces date back to the beginning of the $20th$ century, and the origin of NLP as a field of study is generally credited to Alan Turing, when he proposed the *Turing Test*. Since then, many researchers worked on the field and developed successful systems, and the interest for NLP has grown steadily in the last years in particular, with the widespread of Neural Networks and deep learning-based approaches, which have been supported by the huge increase in computational capabilities of modern computers.
Natural Language Processing systems are built for numerous types of applications, those may include machine translation, information extraction, text analysis and many more.

Word embedding refers to a collection of techniques used in natural language processing. The basic idea behind this method is converting words into vectors of real numbers. The term *embedding* derives from the mathematical transformation, called embedding, that is applied on an object x that is mapped onto another object y. In our case, x and y are two vectors that belong to different vector spaces of size N and M respectively, with $M < N$. In fact, the embedding operation is employed to significantly reduce the dimension of the vector space while at the same time preserving the knowledge of the original object x.

$$f : \mathbf{x} \rightarrow \mathbf{y}, \qquad \mathbf{x} = \{x_1, x_2, ..., x_N\}, \;\; \mathbf{y} = \{y_1, y_2, ..., y_M\}$$

There are different methods for implementing the mapping function, in Section 4.5 we explore a possible algorithm for efficiently learning word vectors.

The idea of transforming words into vectors of numbers is not novel; other techniques existed before word embeddings and are currently being used. However, word embeddings have several benefits over traditional word vectorization techniques, and are able to efficiently capture semantic and syntactic information. For this reason they are of particular interest for researchers and it has been demonstrated that the use of this procedure can boost the performances of many NLP applications [97, 98, 99, 100].

4.1 Motivation

Let us consider as an example image and audio processing systems. These are both based on high-dimensional dense data encoded as vectors that represent pixel intensities or spectral density coefficients for audio data. In contrast, traditional natural language processing systems usually treat words as atomic symbols, each of them having a precise and distinct representation that is different from the others. An example of such a technique is *one-hot encoding*: a word is represented as a vector of dimension N, where all elements of the vector are zero except for the one corresponding to the specific word that is set to one. Here N is the size of the vocabulary, that means, the number of distinct word that are being processed.

A example of a technique that is based on one-hot encoding is term frequency-inverse document frequency (TF-IDF, [101]). This is a numerical statistic that indicates the importance of a word within a document corpus, and it is generally used as a weight for text-mining techniques. TF-IDF counts the frequency of a word in a set of documents weighted by the number of documents in which the word appears, and it is possible to represent a generic document as a vector of TF-IDF weights, one for each word that composes it. In this approach, weights are computed using words coded as one-hot vectors.

Techniques like one-hot encoding suffer from two major issues that arise from the codification itself:

1. Two words that are similar in their context or in their use, for example “dog” and “pet”, are going to be represented as different vectors. A NLP model that analyzes these two words can leverage very little of what it has learned about “dog” when it encounters “pet”.

2. Representing words as unique and distinct ids leads to extreme data sparsity. For one-hot encoding, word vectors have the same dimensions as the number of distinct words. This is absolutely not scalable since best performing NLP applications may work even with millions of distinct words.

The use of word embeddings effectively addresses both the problems described above. In fact, words are no longer represented as unique tokens, but by dense

vector of continuous values of much smaller dimensionality. With dense is intended that each element of the vector is a continuous value in a certain range, instead of having most of them set to zero. This way, it is possible to represent a significantly higher number of words while at the same time reducing the vector size. Regarding the first issue, word embedding are learned in a way such that similar words are mapped to nearby points in the vector space, so we know that if the distance between two vectors is small, the words that they represent are also similar. Similarity between words may have different meanings. Two words are similar if they are synonyms, if they represent the same concept, or if they are used in similar contexts.

Furthermore, since a word vector stores hundreds of continuous values, a single sample carries a lot more information that can be exploited by a machine learning application as features.

4.2 History

Word embedding is a relatively novel technique in the field of natural language processing, but the idea of representing words as numeric vectors dates back to the 1960s with the development of the Vector Space Model (VSM) for information retrieval.

Vector Space Model is a model for representing objects as vector of identifiers. In the case of NLP, the objects are the textual documents composing the corpus, and the identifiers are real numbers. A vector space of dimension N is defined by a set of N linearly independent basis vectors. They are independent in the sense that knowing a vector's value based on one dimension does not say anything about its value on other dimensions. For text processing, each one-hot encoded word is a basis vector of the space, so that its dimension corresponds to the total number of distinct words in the corpus. A group of word, which we refer to as a document, can be represented as a vector V in the N-dimensional space as a combination of the basis vectors. Depending on the technique used for combining the words, vectors will have different values, hence different representations in the space. Possible methods for creating document vectors are binary weights and TF-IDF.

Vector space model depends to some extent on the *Distributional Hypothesis*, which states that words that appear in same contexts also share semantic meaning [102]. Although the hypothesis originated in the field of linguistics [103], it later received attention in cognitive science, in particular regarding studies on word use [104].
However, representing word documents in the vector space can lead to extremely high-dimensional vectors, which constitutes a problem both for model interpretability and for computational complexity: this led to the development of dimensionality reduction techniques such as Latent Semantic Analysis [105]. In the early 2000s,

Bengio et al. [106] proposed a neural probabilistic language model for learning distributed representation of words. Word embeddings then become one of the hot topics for NLP after 2010, when significant advances made it possible to reduce the training speed of the models while at the same time improving their quality.

At present, most of the research on this field is based on neural networks. Many pre-trained embeddings model can be found in literature, such as Word2Vec from Mikolov et al. [35, 107], who were able to learn meaningful word vectors considerably faster with respect to other approaches. They used a simple neural network with one hidden layer and one output layer to efficiently train word embeddings in the order of millions. Other models that are worth of note are GloVe from Pennigton et al. [43], which is based on word co-occurrence statistics rather than neural networks; finally, FastText from Facebook research [108, 109] is based on the approach of Word2Vec.

4.3 Using Word Embeddings

When it comes to using word embeddings, there are two possible choices:

1. Learn word embeddings;

2. Use pre-trained models.

The first case implies choosing an approach, retrieving a considerable amount of raw textual data, and properly tuning the algorithm and its parameters, hence it requires much more time both for designing the process and for learning the actual vectors. On the other hand, using a pre-trained model avoids the need to carefully design and tune the algorithm, yet existing ones in literature may be too general of the language, so if the intention is to apply them on a specific field of for a very precise task, such models may not be particularly well-suited, and it is recommended to learn ad-hoc embeddings which may perform better than other models.

Regarding the more practical question of how to use word embeddings that depends on the particular goal of the application, but generally the real valued vector components of the embeddings are used as features to create a feature vector, to be used in a machine learning approach. If needed, it is possible to extend the feature vector with additional textual or non-textual information, depending on the context. The final vectors could then be processed by a convolution neural network that combines words and phrases together to accomplish a specific task, but this is just an example of all the possible applications.

4.4 Learning Word Embeddings

Word embeddings are usually learned from a huge corpus of unstructured textual data, in the order of billions of words. Such great numbers are required to ensure that the resulting word vectors are effective and useful, meaning that they successfully capture syntactic and semantic similarities between words. Common sources of text data include Wikipedia dumps, Google news dataset, UMBC webBase corpus, Statistical Machine Translation website[1], Polyglot project[2] and others.

Learning word embeddings may be particularly useful, or in some cases required, when dealing with field-specific text data, for which common text sources such as Wikipedia may not cover the whole span of use of certain words, or may cover it only in part (e.g. without capturing meaningful relationships). For example, Social Media Sites feature a use of English language that is slightly different from the one used in Wikipedia, containing abbreviations, slang terms, and a higher rate of errors. While errors should be tackled separately and with other approaches, learning word embeddings from text extracted from Social Media is likely to improve the quality of the vectors. So, if it is known in advance the field of application of the word embeddings, it is recommended to learn them directly on data from that field. However, field-specific text data may not be as easily available as more general text such as Wikipedia, in particular when such a large amount is required. In those cases, it is possible to extend an already trained model using only the field-specific text data of interest.

4.4.1 Methods

Method for learning word embeddings include neural networks [35], dimensionality reduction on word co-occurrence matrix [110, 111, 112], probabilistic models [114], and representing words based on the context in which they appear [118]. We briefly introduce and describe those approaches.

Neural networks word vectors are learned by training a simple neural network with 3 layers (input, hidden, and output) to perform a certain task, for example text classification. During training, the network learns and updates the weights oh the hidden layer, that once finished are the word embeddings themselves.

Co-occurrence matrix and dimensionality reduction The whole document corpus is analyzed to compute co-occurrence frequencies by counting how

[1] http://statmt.org/

[2] https://sites.google.com/site/rmyeid/projects/polyglot

many times each context word appears after a sequence of other words. Since the co-occurence matrix is vocabulary size-dependent, it is not really tractable for large vocabularies, so a dimensionality reduction technique as PCA is required.

Probabilistic models Objects co-occurrence statistics are used as a source of information about similarity. The mapping function is constructed so that a pair of objects that are embedded as two nearby points in the map have a higher statistical interaction than a pair that is embedded as two distant points.

Context-based representation Each word is associated with a sparse high-dimensional vector capturing the contexts in which it occurs. Relationships for the whole language are represented by a sparse matrix of words and contexts, where each element indicates the strength of the association between word i and context j.

Novel methods based on neural networks seem to be the most efficient and successful. There are many studies and works in literature that use this approach, and we are going to explore one of them in the next section.

The dimension of the reduced vector space, which we previously called M, may depend to some extent on the context of application of the word embeddings, or, especially for neural networks-based approaches, on more technical constraints such as computation capacity and memory availability. In fact, training a neural network on such a huge corpus of text data is highly computationally expensive, and before the efficient approach proposed by Mikolov et al. [35, 107] it was not possible to learn meaningful word vectors with a dimension greater than 100/150. Nowadays, the common dimension used for word embeddings is 300, although current algorithms and computers allow to have a significant bigger one. The choice of 300 appears to be empirical, as it seems to outperform other values. Those findings were documented by Landauer and Dumais [115], who examined the relationship between vector dimension and their performances, although they conducted their study using LSA and dimensionality reduction techniques rather than neural networks. Figure 4.1 reports their original findings, highlighting the performance peak at around 300 dimensions. This behavior may be explained by the fact that too few parameters make the model incapable of fitting the signal, while using too many parameters may result in overfitting.

4.5 Word2vec

Word2vec is a computationally-efficient predictive model for learning word embeddings from textual data. It provides two algorithms: the Continuous Bag of

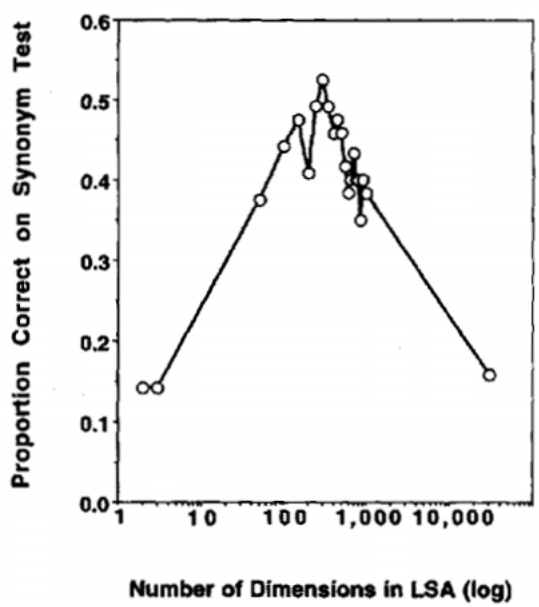

Figure 4.1. Relationship between word embeddings dimensionality and their performance, extracted from the original paper of Landauer and Dumais [115]. As we can see using too many dimensions produces results similar to using too few.

Words (CBOW) and the Skip-Gram model. Algorithmically, these two models are quite similar, with the difference that CBOW predicts target words (e.g. "squirrel") given their context ("the dog barks at the"), whereas the skip-gram does the inverse, it predicts source context words from the target words. By treating the entire context as a single observation, CBOW smoothes over a lot of the distributional information, which is useful for smaller dataset. Conversely, skip-gram is preferable for larger datasets, because it treats each context-target word pair as a new observation. We will see in detail the skip-gram model.

4.5.1 Skip-Gram Model

Word2vec uses a trick that is not infrequent in machine learning. It trains a simple neural network with one hidden layer to perform a specific task, but it does not use that neural network for the task it was trained on. Instead, what we are interested in are the learned weights of the hidden layer. These weights correspond to the word vectors we are looking for. Thus, although word embeddings are often associated with many deep learning applications, the neural network used to learn

them is actually pretty simple.

The training objective of the Skip-gram model is to find word representations that are useful for predicting the surrounding words in a given context, for example a sentence or a document. Given a sequence of training words $w_1, w_2, w_3, ..., w_T$, the objective of the model is to maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} log\, p(w_{t+j}|w_t)$$

Where c represents the size of the context window. The probability $p(w_{t+j}|w_t)$ is defined by means of the softmax function (see Section 4.5.1 for more details).

The probabilities computed by the network indicate the likelihood of having that term in the same context of the input word. Referring to the previous example, our target word is "squirrel", and context words are "the dog barks at the". Probabilities computed by the networks are going to be higher for context words than for other unrelated words, such as "phone", which are in the vocabulary but never appear in the same context with "squirrel".

If two words have similar context, for example "food" and "pizza", the network should output similar results when we use these words as input. We will see that in order for this to happen, their corresponding vector representations, or word embeddings, also need to be similar. The neural network is thus motivated to learn similar embeddings for words that often appear together, or that share semantic meaning.

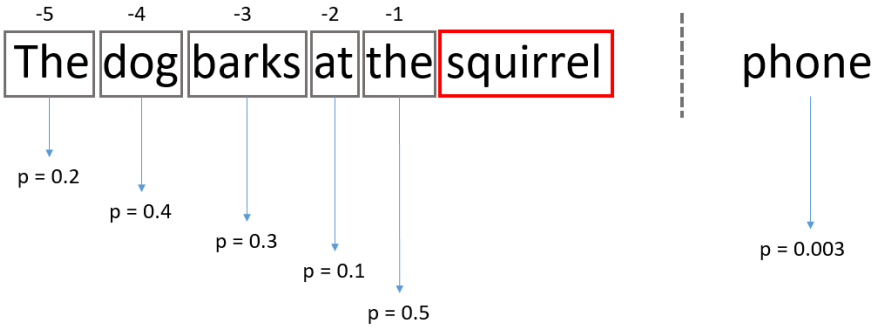

Figure 4.2. Output of the neural network given the input word "squirrel". Probabilities of context words are higher than those of other words.

The neural network is trained by feeding word pairs extracted from the training document corpus. Figure 4.3 shows an example of how training samples are created starting from a collection of raw textual data. Each pair is a single training sample.

By feeding a very large number of training samples to the network, it is going to learn the statistics of word co-occurrences and word similarities. At the end of the training phase, the network is likely to output higher probabilities for word related to the target one, just as in Figure 4.2.

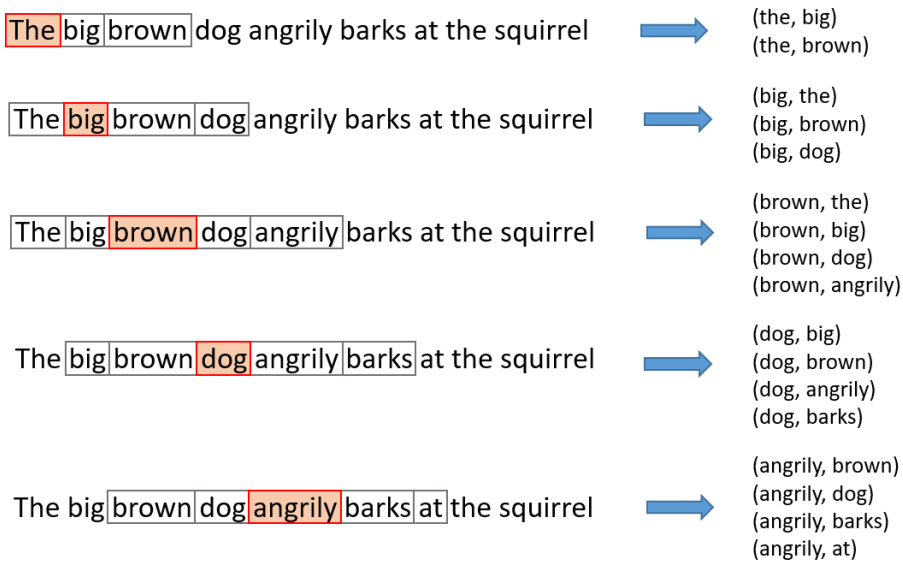

Figure 4.3. Extraction of training samples from documents using a sliding window with size 2.

Neural Network Structure

We cannot feed a word in the neural network just as it is, we first need to represent it as a numerical vector. The easiest way to do this is using one-hot encoding: words are represented as binary vectors having the same size of the vocabulary; all vector values are zeros, except for the one corresponding to the word position. Let suppose that we use a vocabulary of 100.000 words. One-hot vectors have the same dimension, with 99.999 elements set to zero. The output of the neural network is again a vector of dimension 100.000, but this time each component stores the probability we are interested in. Figure 4.4 illustrates the structure of the network.

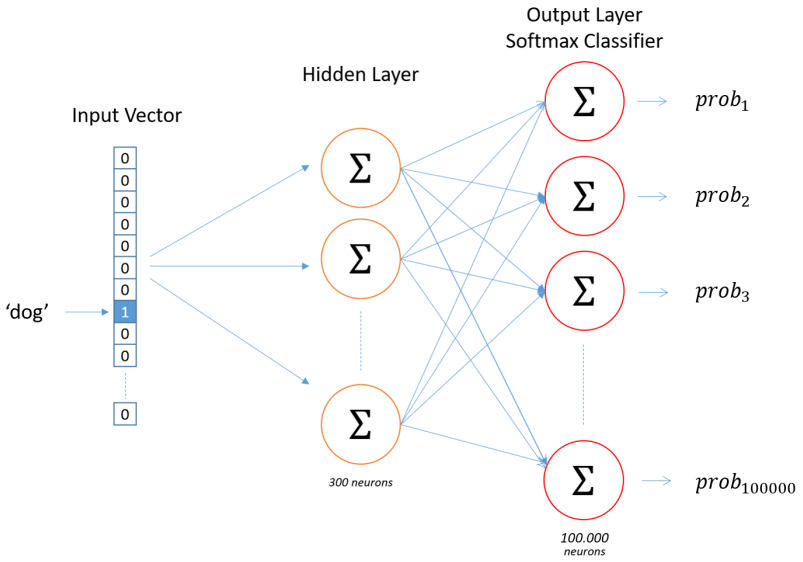

Figure 4.4. Neural network structure.

Hidden Layer

Figure 4.4 shows that the hidden layer counts 300 neurons, which will also be the dimension of the learned word embeddings. The number of neurons is a hyperparameter that needs to be tuned for the application, but experimental results have proved that 300 is an appropriate size. See Section 4.1 for more details.

The hidden layer is represented by a weight matrix with 100.000 rows (the size of the vocabulary) and 300 columns (number of neurons); all weights are initialized to small random values. When a one-hot vector is fed to the neural network, what really happens is a matrix multiplication between the word vector and the weight matrix. Because of the properties of this operation on matrices, the result is the 300-dimensional matrix row corresponding to the position of the value 1 in the one-hot vector. This vector is the word embedding itself, but since the network is still learning the weights, it may not store the final features we are looking for, and they are very likely to be updated afterwards.
There is no activation function on the hidden layer, its output is simply the i-th row of the weight matrix, which is then fed to the output layer.

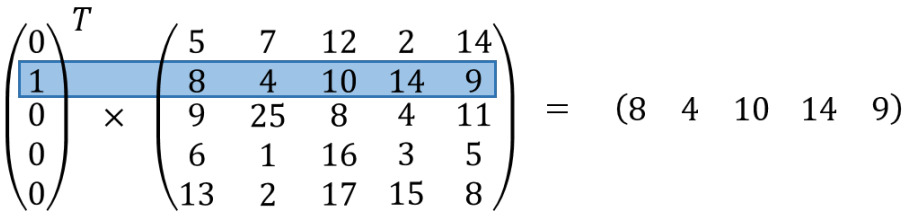

Figure 4.5. The result of a multiplication between a $1 \times N$ vector and a $N \times M$ matrix results in a $1 \times M$ vector. This is the i-th matrix row corresponding to the input vector element with value 1, which is at position i in the vector. In our case, $N = 100.000$ and $M = 300$.

Output Layer

The output layer is only used in the training phase of the neural network. Once this has ended, it is not needed anymore. It is composed of a number of neurons that is equal to the size of the vocabulary. Each of those neuron uses the word vector computed in the hidden layer to produce its output using a softmax regression classifier. We will not explain in detail the softmax function, since it goes beyond the purpose of this thesis. Basically, each unit computes the dot-product between its own weight vector and the word vector, then applies $e^{(x)}$ to the result. The probability is then normalized by dividing it by the sum of all the 100.000 scores, so that the final results of each neuron adds up to 1. Given a word w_t (for target) and a context h (for history), the probability score is computed as follows:

$$P(w_t|h) = softmax(score(w_t, h)) = \frac{exp\{score(w_t, h)\}}{\sum_{Word\, w'\, in\, Vocab} exp\{score(w', h)\}}$$

Where with $score()$ we have indicated the dot-product. An example of computing the output probability for a word is shown in Figure 4.6. After computing the output probabilities, there is one last thing to do: update the weight matrix of the hidden layer. This is done with backpropagation of the error.

Backpropagation

Backpropagation consists in calculating the output errors and modifying the weights of the network accordingly. It is commonly used with the gradient descent optimization algorithm which adjusts the weight of the neuron by calculating the gradient of the loss function. During training, a number of input-context word pairs are fed to the network. Both are represented as one-hot vectors, the first is used to compute the output probabilities, which are then compared to the second one in order to compute the error, using a loss function.

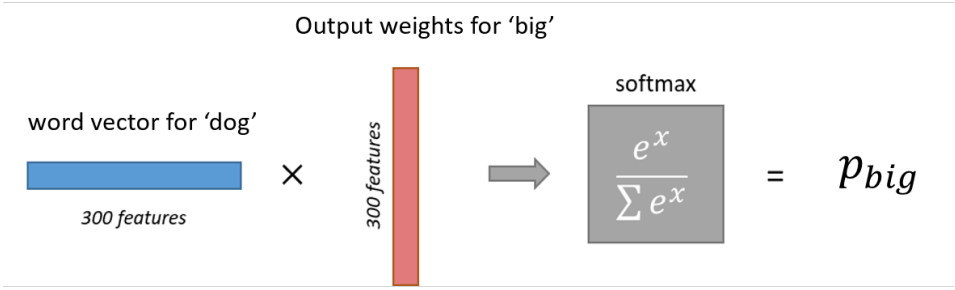

Figure 4.6. Example of computation of the output probability of the word "big" given the input word "dog". Each output neuron has its own weight vector that uses for computing the dot-product with the word embedding coming from the hidden layer. This process is repeated for each neuron (= for each word in the vocabulary).

For the skip-gram model, the loss function is defined as the the *Maximum Likelihood* (ML) principle. The neural network is trained using ML to maximize the probability of choosing a word given its context:

$$J_{ML} = logP(w_t|h) = score(w_t, h) - log\left(\sum_{Word\, w'\, in\, Vocab} exp\left\{score(w', h)\right\}\right)$$

Adjustments to the weights of the hidden layer are computed starting from the formula above and applying the gradient descent algorithm, thus calculating the derivative.

Iterations are repeated until the maximum number of epochs is reached, or the output error is under a specific threshold.

However, this algorithm suffers from two major issues. First, word vectors are limited by their inability to represent idiomatic phrases that are not compositions of the individual words. As an example, the authors use the pair "Boston Globe", which is a newspaper. The words "Boston" and "Globe" have a completely different meaning with respect to their combination.

The second issue concerns the time required for training. When learning word vectors from a sizeable corpus, the neural network tends to become huge. In the simple architecture that we considered in the last sections, we have 100.000 distinct words and 300-dimensional embeddings. This means that there is a total of $100.000 \times 300 = 30$ million weights both for the hidden layer and the output layer.

Such architectures are often trained on a very large document corpus, such as Wikipedia dumps. Those documents can easy reach the order of millions of words,

so it becomes clear that despite being fairly intuitive and straightforward, this approach becomes prohibitive when it comes to real learning scenarios.

The authors of word2vec addressed those issues in their second paper [107]. Three simple extensions of the original algorithm have been proposed, we list them below and briefly explain the intuition that motivated those innovations.

- Treating word and phrases that are commonly found together as single samples;
- Subsampling frequent words to lower down the number of training samples;
- Modifying the optimization objective with a method called *Negative Sampling*: each sample only leads to the update of a small percentage of weights.

Word and Phrases

The simple composition of two words can have a meaning that is completely different and unrelated to the original words it is composed of, so it is reasonable to trait those cases as single training samples for the model.

To learn vector representations for phrases, it is first needed to find words that frequently appear together, but rarely in other contexts. For example, "Spanish Airlines" and "Mount Everest" are replaced with a special token in the training set, while common bigrams such as "this is" remain unchanged. Less meaningful bigrams are only ignored because of memory limits. The proposed approach only considers combinations of two words, but it can be run multiple times to extract phrases.

The model was trained on a corpus of 100 billion total words from part of Google news dataset, and it learned word vectors for 3 million words and phrases. It was made publicly available by Google[3]. Experimental result show that accuracy and quality of learned word vectors significantly scale with the size of the dataset.

Subsampling of Frequent Words

Figure 4.3 illustrates how training samples are extracted from raw text. The method is pretty simple and straightforward, although it highlights two problems with very common words, such as "the":

1. When "the" appears as context word, for example ("dog", "the"), it provides us with almost no information about the context of "dog";

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

2. Since it is extremely frequent, there will be a huge number of samples featuring "the" as input word, way more than needed to learn a meaningful vector for "the".

Words with such characteristics are referred to as *stopwords* and are often ignored in many NLP applications.

Word2vec does not completely ignore stopwords, but implements a subsampling method to address the problems described above. For each word encountered in the training text, the model computes the probability of keeping it, which is based on its frequency:

$$P(w_i) = \left(\sqrt{\frac{z(w_i)}{0.001}} + 1\right) \cdot \frac{0.001}{z(w_i)}$$

Where $z(w_i)$ is the fraction of total word in the corpus that are the word w_i, and 0.001 is a parameter called *sampling rate*, and it regulates the amount of subsampling that is performed. Figure 4.7 shows how word frequency and sampling rates affect the probability of keeping a word. Given a word frequency (e.g. 0.05), smaller rates result in lower probabilities, hence words are more likely to be discarded.

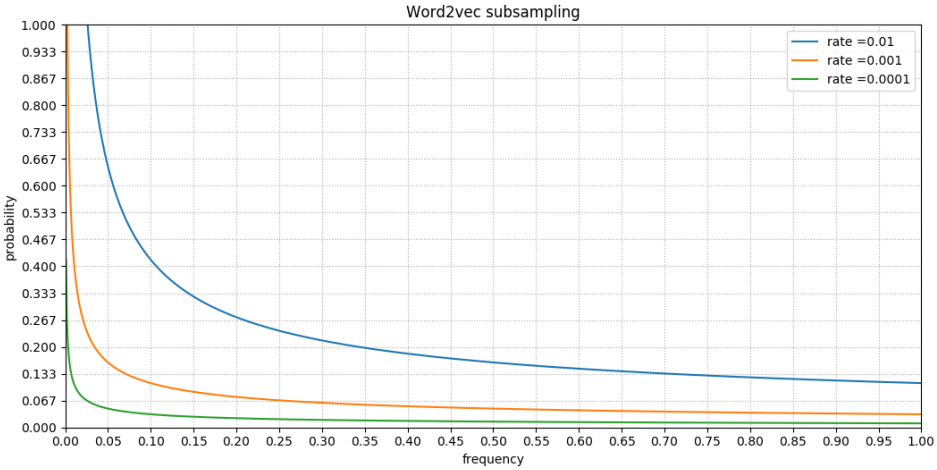

Figure 4.7. The three curves represent the probabilities of keeping a word in function of different sampling rates. For a fixed word frequency, smaller rates correspond to lower probabilities.

Removing a word from the corpus effectively solves both the problems listed above:

1. "The" will not appear in the context of the other words;
2. There will be N fewer training samples for "the", with N corresponding to the window size.

Although the formula was chosen heuristically, as well as the threshold value, this method can significantly accelerate learning (around 2x - 10x) and even improve the accuracy of the vectors of rare words.

Negative Sampling

Typically, when training a neural network, each sample causes a small adjustments to all the weights of the matrix. This is very expensive, since as we have seen the number of weights can grow to the order of billions, as well as the number of training samples. With negative sampling, a single input-context word pair only accounts for a small percentage of weights update.

Each pair is composed by an input word, which is fed to the network, and a target word, which is compared to the network output to compute and backpropagate the error. Both words are represented as one-hot vectors, so one element is set to one while the remaining 99.999 to zero. With negative sampling, the target vector only has a small number of elements set to zero, and we refer to those elements as *negative* words or samples. Only the weights of those words, together with the one of the input word, will be updated, resulting in a significant speedup of training time.

Mathematically, for each sample the model maximizes

$$log\,\sigma(v'^{\top}_{w_O} v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[log\,\sigma(-v'^{\top}_{w_i} v_{w_I}) \right]$$

This replaces every $log\,P(w_O|w_I)$ occurrence in the original Skip-gram objective (see Section 4.5.1). The task is to distinguish the target word w_O from draws from the noise distribution $P_n(w)$ using logistic regression, having k noise samples for each data sample. Experiments revealed that for small datasets accurate k values are in the range $5-20$, whereas for large datasets values values in $2-5$ are sufficient.

Negative samples are chosen using a unigram distribution: words are chosen as negative samples according to their frequency, with more frequent words being more likely to be selected:

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n} \left(f(w_i)^{3/4}\right)}$$

The choice of raising the frequency to the 3/4 power is empirical, it simply outperformed other functions.

4.6 Visualizing Word Embeddings

As mentioned, an interesting property about word embeddings is context similarity: representations of words that are similar in the language will be close in the vector space. Here, the word *similar* can have different meanings. Similar words may be synonyms, word pairs frequently occurring together, proper names of states or capitals, or even same verbal forms (e.g. past simple). Visualizing learned word vectors is then a fast and immediate way of verifying their quality.
However, we cannot represent a vector with hundreds, if not thousands, of dimensions straightforwardly; we first need to create a two-dimensional representation using a dimensionality reduction technique. A possible solution for this problem is t-SNE.

4.6.1 t-SNE

t-Distributed Stochastic Neighbor Embedding, or t-SNE [117], is a dimensionality reduction technique that is particularly well-suited for visualizing high-dimensional data by reducing vectors to two dimensions.

Dimensionality reduction techniques generally convert high-dimensional datasets $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ into two-dimensional data $\mathcal{Y} = \{y_1, y_2, ..., y_n\}$ so that they can be graphically visualized in a scatterplot. These techniques try to preserve as much information as possible when creating the low-dimensional map, that is, $\mathcal{Y}$. t-SNE successfully captures the structure of the high-dimensional data, also revealing the presence of clusters at several scales. For this reason it is particularly well-suited for the task.

The original stochastic neighbor embedding technique was presented by Hinton and Roweis [116]. Van der Maaten and Hinton [117] then advanced t-SNE, which addresses some issues of the original technique.

Stochastic Neighbor Embedding

Stochastic Neighbor Embedding converts the Euclidean distance between two datapoints x_i and x_j into conditional probabilities that represent similarities. The similarity between x_i and x_j is the conditional probability, denoted as $p_{j|i}$, that x_i would pick x_j if neighbors were picked in proportion of their probability density under a Gaussian centered at x_i. The probability will be high for points that are close in the high-dimensional space, and infinitesimal for points that are distant. Mathematically, $p_{i|j}$ is computed as follows:

$$p_{j|i} = \frac{exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-\|x_i - x_k\|^2/2\sigma_i^2)}$$

In the above formula, σ_i represents the variance of the Gaussian that is centered on point x_i. Datapoints x_i and x_j are going to be mapped to their low-dimensional correspondents y_i and y_j. It is possible to compute the conditional probability, $q_{j|i}$, in a similar way:

$$q_{j|i} = \frac{exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} exp(-\|y_i - y_k\|^2)}$$

In this case σ_i does not appear because it is set to $\frac{1}{\sqrt{2}}$. Setting σ to a different value with respect to $p_{j|i}$ will only result in a rescaled version of the final map. If y_i and y_j correctly model their high-dimensional counterparts, $p_{j|i}$ and $q_{j|i}$ will be equal. SNE aims at finding a mapping that minimizes the mismatch between $p_{j|i}$ and $q_{j|i}$. In particular, SNE minimizes the sum of Kullback-Leibler divergences over all datapoints using an approach based on gradient descent. The cost function C is given by:

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

P_i and Q_i represent the conditional probability distribution over all datapoints given x_i and y_i, respectively.

t-Distributed Stochastic Neighbor Embedding

Although SNE constructs fairly good representations, it has two major issues: the cost function is rather difficult to optimize, resulting in a high computational complexity, and second, the approach suffers from the so called "crowding problem". The authors address the first issue by implementing a symmetrized version of original SNE cost function that features a simpler gradient.

Symmetric SNE Instead of minimizing the sum of the Kullback-Leibler divergences between $p_{j|i}$ and $q_{j|i}$, it is possible to minimize a single Kullback-Leibler divergence between a joint porbability distribution P in the high-dimensional space and a joint probability ditribution Q in the low-dimensional space. This is referred to as *symmetric SNE*, because for all i and j is valid the property $p_{j|i} = p_{i|j}$ and $q_{j|i} = q_{i|j}$. New cost function will be:

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$$

The crowding problem The problem is defined as follows: the area of the two-dimensional map available for accomodating moderately distant datapoints will not be large enough compared with the area available to accomodate nearby points.

In other words, if we want to accurately model the nearby points, those that are at a moderate distance will have to be placed too much far away in the map. This problem is not specific of SNE, but it also occurs with other techniques.
The crowding problem is solved using a t-Student distribution rather than a Gaussian in the low-dimensional space.

4.6.2 Implementing t-SNE

We report in Table 4.1 the pseudo-code of t-SNE as presented by the authors in the original paper [117]. It is also possible to download several implementations of the algorithm in numerous languages[4]. In our study, we used Python implementation available on the website.

Simple version of t-Distributed Stochastic Neighbor Embedding
Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: num iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
| compute pairwise affinities $p_{j\|i}$ with perplexity *Perp*
| set $p_{ij} = \frac{p_{j\|i} + p_{i\|j}}{2n}$
| sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0,10^{-4}I)$
| **for** *t=1* **to** T **do**
| | compute low-dimensional affinities $q_{j\|i}$
| | compute gradient $\frac{\partial C}{\partial \mathcal{Y}}$
| | set $\mathcal{Y}^t = \mathcal{Y}^{(t-1)} + \eta \frac{\partial C}{\partial \mathcal{Y}} + \alpha(t)(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)})$
| **end**
end

Table 4.1. Original pseudocode for the t-SNE algorithm.

Although the existing t-SNE implementations are quite intuitive and straightforward to use, they still require some tweaks for properly formatting the data we want to visualize. It is possible to avoid doing this using the online TensorBoard Embedding Projector[5], which handles all technical aspects of the visualization and

[4] https://lvdmaaten.github.io/tsne/

[5] http://projector.tensorflow.org/

allows to explore the two or three-dimensional map that it creates.

4.6.3 Visualizing FastText Embeddings

We run t-SNE on the first 2000 words contained in the pre-trained word embeddings from FastText that we use in our research. The result of the execution is an interactive graph that shows all the words in a two-dimensional scatterplot, together with their labels. We don nott visualize the whole FastText vocabulary, which is composed of one million words, because t-SNE would take too much time. Moreover, 2000 words are sufficient to properly show the context similarity property of the embeddings. The plot is showed in Figure 4.8. The high number of points and labels makes it difficult to read the plot, however it is possible to zoom and explore it at a closer distance, selecting only a part of it.

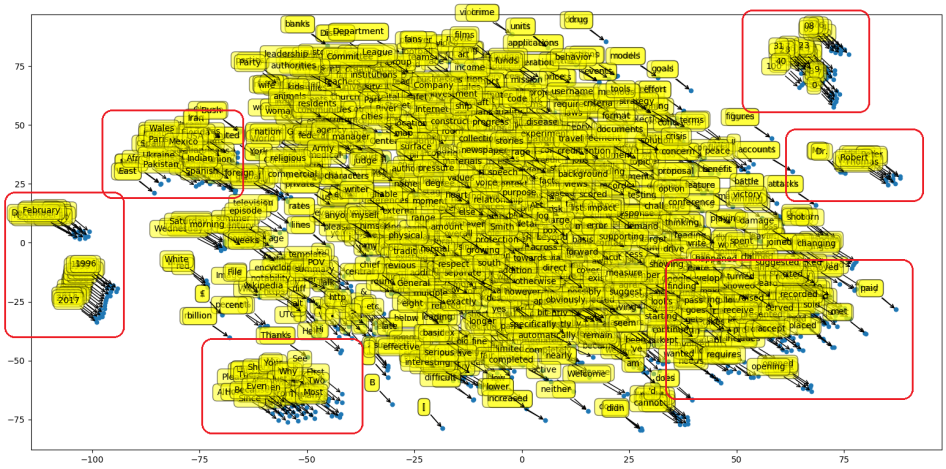

Figure 4.8. Word embeddings for the first 2000 words of the FastText dataset. Even though the plot is rather chaotic, it is clearly possible to identify some evident word clusters.

Anyway, it is still possible to distinguish small clusters of similar words, which are highlighted by a red box in the figure. Those clusters are composed by numbers, months, years, stopwords, proper names and abbreviations, geographic locations, verbs in different forms, days of the week and measures of time. Those are just some of all the groups of similar words that can be seen in the plot. Figures 4.9, 4.10, and 4.11 illustrate some of the above mentioned word clusters and show some interesting semantic properties they captured.

If a higher number of word was plotted, the number of clusters, together with their size, would increase accordingly.

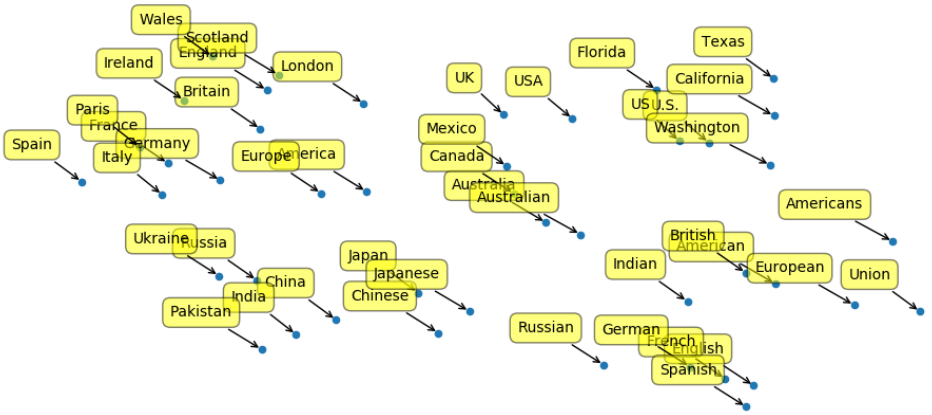

Figure 4.9. Word embeddings of states, capitals, and nationality adjectives. We can also see that "European" and "Union" are close each other and to other similar words., although Union is not related to a geographic location. This is an example of treating word pairs as single training samples.

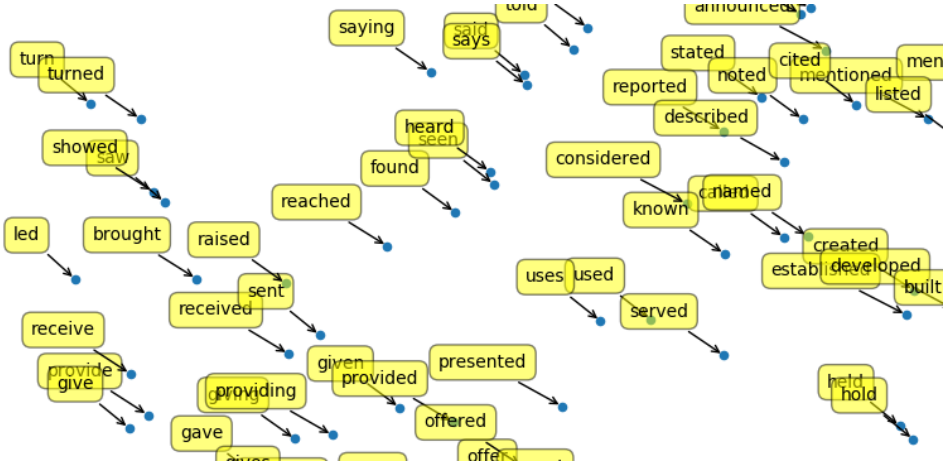

Figure 4.10. Word embeddings for verbs. Most of then are in past tense forms, and some of them are also close to other forms, such as simple present and present continuous of the same verb. This is true only for a small part of the verbs because we only plotted the first 2000 words of the dictionary.

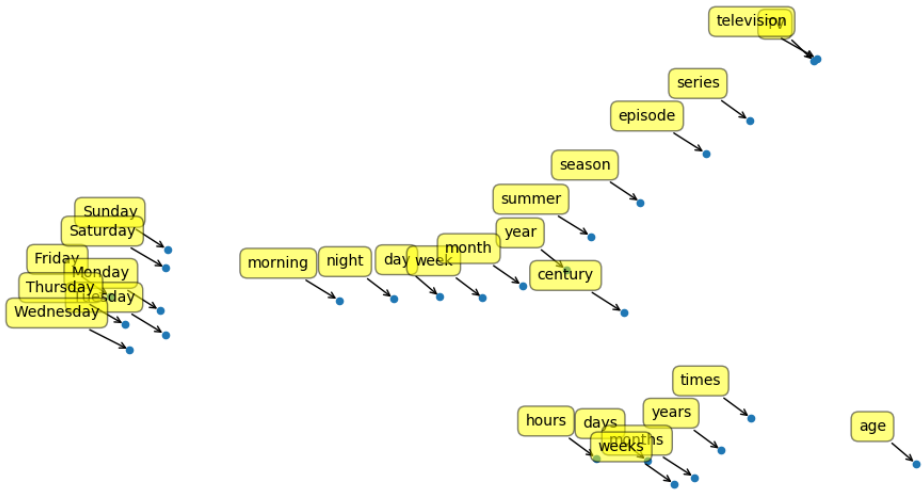

Figure 4.11. Word embeddings for days of the week and various time measures and periods. It is interesting to notice that as we move from left to right, the time measure increases; from morning and night to century. Moreover, the word season is close both to words related to time and to words related to TV shows, demonstrating that the corresponding embedding successfully captured both semantic meanings of that word.

Chapter 5

Approach

In this chapter we present in detail the approach we adopt to achieve our task, that is, predicting personality from tweets. To do that, we employ a machine learning algorithm to train a predictive model on a set of labeled training samples extracted from a dataset of Facebook status updates, which has been collected for scientific research. We refer to this dataset as the *Gold Standard*. Section 5.1 describes it in detail and provides some statistics of the data that it contains.

Once we have successfully trained a model, we can then test its predictive power on unknown individuals. To do this, we set up an experiment involving a few Twitter users who gave their consensus to participate in the test. We crawl the Twitter API to download all the tweets available for those users, which we clean to enhance their similarity to Facebook status updates. We then preprocess and transform each tweet the same way as we did for status updates when training the model. Finally, we feed each tweet to the model which computes a numeric score. We aggregate the scores for all the tweets of the same user to compute its personality. At the same time, we ask the participants to fill a personality questionnaire to extract ground truth personality scores, which we compare to those generated by the machine learning model to assess its accuracy. We report a general overview of the whole application in Figure 5.1.

We base our approach on *Transfer Learning*, which means that we learn a model on a domain to subsequently test it on a different one. In fact, the gold standard used to train the model contains status updates from Facebook, while our intention is to predict personality of Twitter users. Although there are a number of similarities between the two (e.g. short posts/tweets containing slang and grammatical errors) they are not the same, so we expect a certain baseline error when switching from a context to the other. In the attempt to smooth out this error we process the tweets more carefully, as we show in the later sections.

Many successful applications of transfer learning can be found in literature.

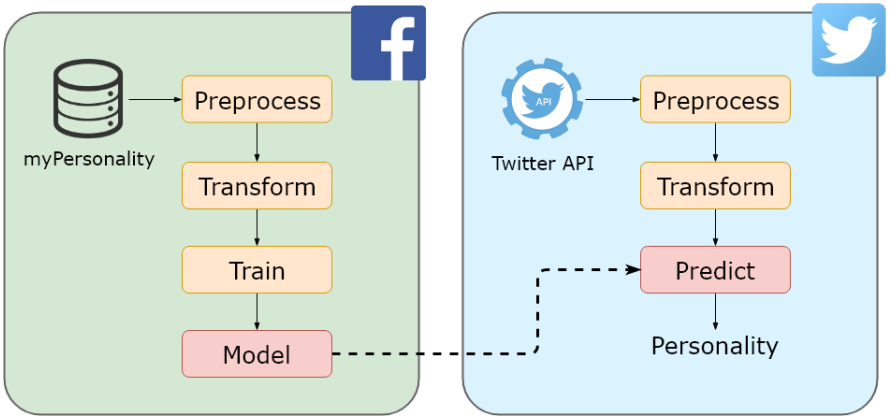

Figure 5.1. Overview of the application. We first train a predictive model with a set of status updates from myPersonality dataset, then we apply it to compute a personality score for users' tweets downloaded using the Twitter API.

For example, both the approaches documented in [120] and [119] are based on that technique to address text classification problems. Although our goal is not classification, those studies demonstrate that applying transfer learning on textual data yields satisfying results.

This chapter is organized as follows: in Section 5.1 and 5.2 we respectively introduce the gold standard that we use to derive a predictive model, and the dataset containing the pre-trained word embeddings; in Section 5.3 we illustrate all the transformations and procedures that we apply to train the model starting from raw text data. Finally, Section 5.4 and 5.5 describe how personality scored are predicted starting from the Twitter handle of a user.

5.1 The Gold Standard

In order to quantitatively test our approach, we rely on the dataset created by the myPersonality project[1] [122]. The dataset has been collected for research purposes by David Stillwell and Michael Kosinski using a Facebook application that administered a personality test and collected a wide range of personal and activity information from Facebook's profile of users under their consent. Participants answered a variable length (20-100 or 300 items) proxy test derived from Costa and McCrae NEO-PI-R [68] and elaborated by the International Personality Item

[1] http://mypersonality.org/wiki/doku.php

Pool (IPIP) project [121]. Section 3.2 explains in detail how the personality questionnaires are administered to a person and how it is possible to derive personality scores from them.
The myPersonality application has been active from 2007 to 2012 and collected a huge amount of data, which is available upon request.

We base our study on a sample of the original myPersonality dataset that has been made publicly available [123]. The gold standard contains 9913 status updates of 250 users (anonymized), annotated with personality traits scores and basic statistics of the users' network, such as number of friends and betweenness. Table 5.1 and Figure 5.2 illustrate some statistics of the dataset. It contains approximately 15,000 distinct terms and 300 stopwords. However, stopwords account for almost half of the total, lowering the predictive power of the model built on this dataset.

Statistics	**Value**	**Lowest**	**Avg**	**Highest**
Total users	250	-	-	-
Total status updates	9913	-	-	-
Status updates per user	-	1	39	223
Total words	146128	-	-	-
Total words after preprocessing	72896	-	-	-
Unique words	15470	-	-	-
Unique words after preprocessing	15185	-	-	-
Word per status update	-	1	14	113
Word per status update after preprocessing	-	0	7	57

Table 5.1. Statistics of the dataset in terms of number of users, status updates and words. We highlight the effect of pre-processing on those measures.

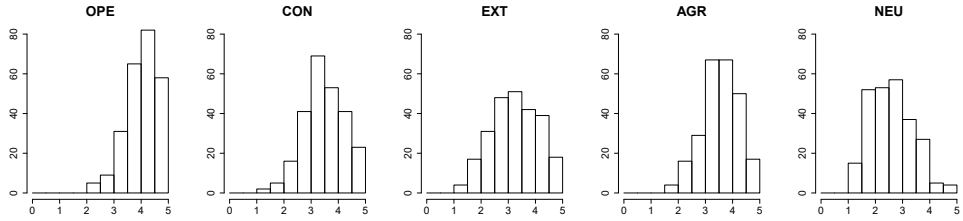

Figure 5.2. Five histograms representing the distributions of the Big5 personality traits of the users available in the gold standard.

trait	OPE	CON	EXT	AGR	NEU
max	5	5	5	5	4.75
min	2.25	1.45	1.33	1.65	1.25
avg	4.0786	3.5229	3.2921	3.6003	2.6272
std	0.5751	0.7402	0.8614	0.6708	0.7768

Table 5.2. Statistics of the personality traits distribution in the gold standard. We report standard deviation, highest, lowest and average value across all 250 users. Where OPE=Openness, CON=Conscientiousness, EXT=Extraversion, AGR=Agreeableness, NEU=Neuroticism.

5.2 Word Embeddings Dataset

The pre-trained word embeddings we use in out study are the english word vectors trained with FastText[2] [33] from Facebook AI Research (FAIR). FastText is an open source library for automatically learning text representation and text classifiers, and the researchers made publicly available many word embeddings models trained in numerous languages. We are only interested in English, so we work with English word vectors trained on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset. The dataset stores 1 million word vectors of dimension 300 and is structured as a dictionary where each line of the file contains an English word followed by its corresponding word embedding. All the values are space separated. Words are ordered by descending frequency, meaning that the first lines store the most common words in the whole document corpus. The word coverage of FastText dataset on the gold standard is 95.08%, but just the first 2% account for more than 80% of the total coverage. Table 5.3 reports as an example the first 30 words of FastText. As we can see, not only words, but also punctuation is included. However, they are all stop-words (very common words that are not informative), and we remove them when processing the textual data, as explained in the following sections.

We also test other word embeddings datasets to assess their overall performance and compare it with the one of FastText. The choice of using the latter comes from empirical results that are documented in Chapter 6.

[2] https://fasttext.cc/docs/en/english-vectors.html

,	the	.	and	of	to
in	a	"	:	)	that
(	is	for	on	*	with
as	it	The	or	was	’
’s	by	from	at	I	this

Table 5.3. Most common words in the corpus used for learning word embeddings, from left to right and from top to bottom. All those words are either stop-words of punctuation. In both cases, they are not informative for the sake of our goal, so they are removed when encountered.

5.3 Model

We illustrate in detail all the steps that we apply to derive a predictive model starting from the status updates that compose the gold standard. Each status update is individually transformed in a corresponding representation in the vector space and together with the personality score becomes a single training sample. All the samples are fed to a machine learning algorithm to derive a model that is capable of predicting personality of unknown users with a certain degree of accuracy. Figure 5.3 summarizes what we just said.

Those steps are repeated five times in total, once for each personality trait, so that when the training phase is completed we have five different predictive models that can be used to predict personality of Twitter users in our transfer learning context.

We now introduce the stages in which our approach is subdivided.

5.3.1 Text Preprocessing

Textual data is likely to contain noise, errors, and less-informative words, which may lower down the quality of the model and harm its overall accuracy, and those issues are even more pronounced when dealing with data extracted from Social Networks, which presents noticeably higher rates of slang terms and spelling and grammar errors. It is then evident that an accurate preprocessing phase is required before exploiting the data to train the model. Text processing steps that we sequentially apply are the following:

Conversion to lowercase : “Today is a #sunny day!” → “today is a #sunny day!”.

Stop-word removal : “today is a #sunny day!” → “today #sunny day!”.

Punctuation removal : “today #sunny day!” → “today sunny day”.

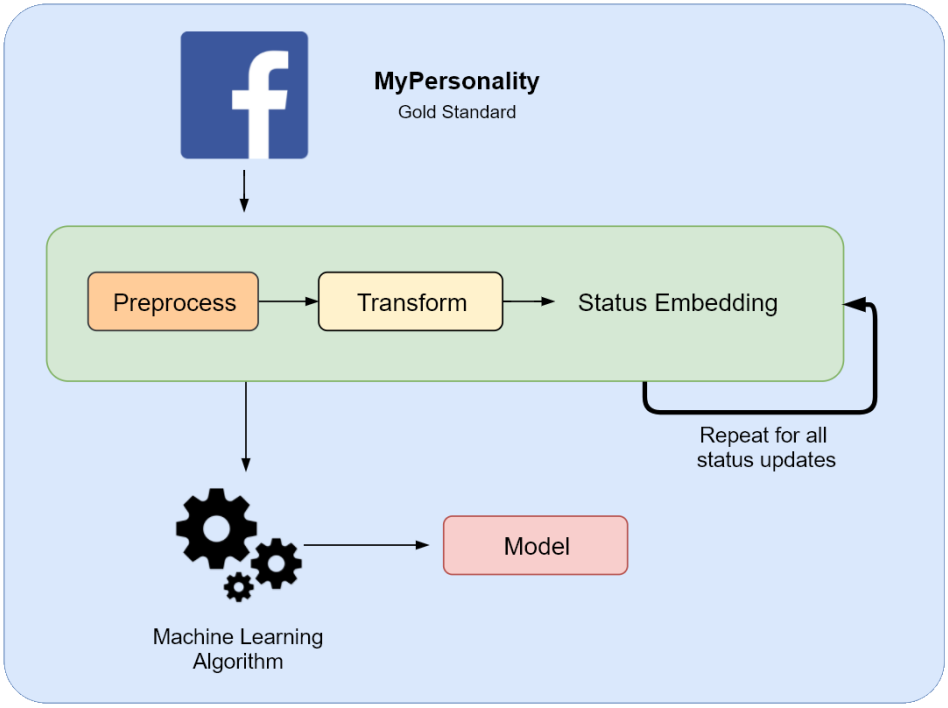

Figure 5.3. Overview of the model training process. Status updates from the gold standard are all processed and transformed the same way, one at a time, to obtain status embeddings. Those are fed to the machine learning algorithm to compute a predictive model for personality.

Tokenization : "today sunny day" → [today] [sunny] [day].

Short post removal : it removes status updates with $n_{tokens} < 3$.

For stop-word removal, we use the list[3] of stop-words provided by the module *CountVectorizer* from *scikit-learn*. We remove status updates that result in less than three tokens, which are roughly 1200 in total. We choose not to apply stemming, because that way we could lose the semantic meaning of a word in its context.

We apply the steps detailed above to each status update in the gold standard, so that raw text is transformed into a structured list of tokens, which is processed in the next stage, namely transformation in the vector space.

[3] https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/feature_extraction/stop_words.py

5.3.2 Text Transformation

While words are easier to read and understand for humans with respect to numbers, the opposite stands for computers. We cannot feed textual data straight to the machine learning algorithm, because it will not understand them. We first need to transform it into numeric signals. To do this, we exploit dense, real-valued representations of words that are also known as word embeddings. We use the word embeddings dataset introduced in Section 5.2 to derive a fixed-length vector from the list of tokens resulting from the preprocessing of raw text from the gold standard. Since status updates have different lengths, the tokens lists resulting from them have different lengths. However, in order to train the machine learning model we need to represent all the training samples with vectors of the same size. In contrast to other approaches based on neural networks [34, 26, 42, 48], we transform status updates into their corresponding representations in the vector space using geometric manipulations of the word vectors, inspired by the work of Li et al. [113].

We lookup each token in the embeddings dictionary to get the corresponding word vector. If it does not exist, we simply ignore that word. After all tokens have been processed, we combine all the word vectors into a single one. Li et al. [113] explore different methods for representing a tweet starting from word embeddings, and test these approaches on two tasks: sentiment analysis and topic classification. We choose to represent status updates with concatenation of maximum, minimum and average. To compute the maximum, we consider for each one of the 300 dimensions the maximum value among all word vectors for that component. In a similar way we also derive minimum and average. Since word vectors from FastText have a dimension of 300, the vector resulting from the concatenation has a dimension of 900. Figure 5.4 illustrates the process of deriving a vector space representation from a list of tokens. The choice of concatenation is verified empirically; in fact, this method appears to be more stable than other approaches, having the least variance in terms of mean squared error

5.3.3 Model Training

We train five different models, one for each personality trait, using word embeddings as the only features. More specifically, we train the models by feeding the 900-dimensional vectors derived from the concatenation together with the personality trait score of the user that wrote the status update, for a total of approximately 8000 training samples (the remaining are removed due to their brevity). Since trait scores are continuous values we choose a regression algorithm, that uses those 8000 samples to learn the relationships between textual features and personality.

We explore three different machine learning algorithms for regression: Linear Regression [124], LASSO [125] and SVM [126].

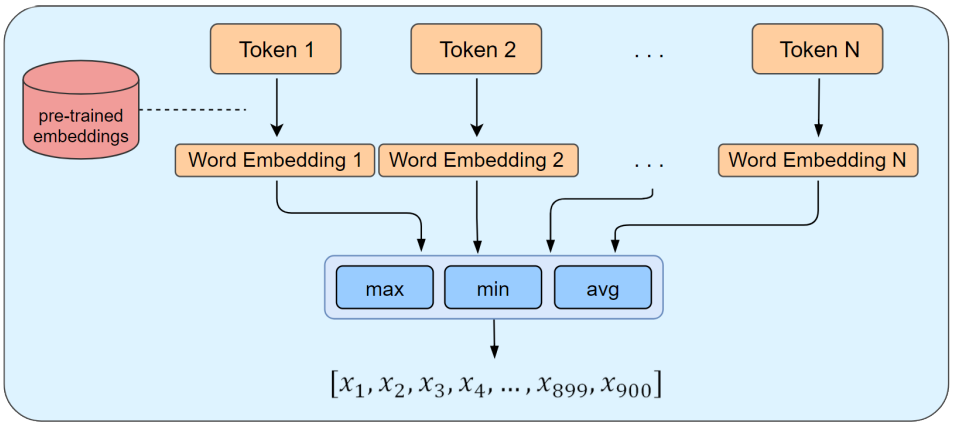

Figure 5.4. Mapping of a post to the corresponding representation in the vector space. For each token we obtain the corresponding word embedding, then we concatenate max, min and avg computed on all word vectors.

Linear Regression

Linear regression is a model used to predict continuous values using a linear combination of a set of features, that in our case are the word vectors. Variables $\mathbf{X}$ and $\mathbf{y}$ are related by an unobserved parameter β_t (where t stands for *true*), the goal is fitting a predictive model to observed data $\mathbf{X}$ and $\mathbf{y}$ by estimating a set of parameters β, so that $\mathbf{y}$ can be expressed by means of $\mathbf{X}$ and β.

$$\mathbf{y} = \beta\mathbf{X} + \epsilon \tag{5.1}$$

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_{900} x_{i900} + \epsilon_i, \qquad i = 1,2,...,N \tag{5.2}$$

The parameter estimation is done with the ordinary least squares method, which minimizes the sum of squared residuals of the predictions, defined as follows:

$$\beta = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \left(\sum \mathbf{x_i} \mathbf{x_i}^\top \right)^{-1} \left(\sum \mathbf{x_i} y_i \right) \tag{5.3}$$

Since β is merely an approximation of β_t, the prediction of $\mathbf{y} = \{y_1, y_2, ...y_N\}$ includes a certain error, which is denoted with ϵ.
Graphically, the underlying relationship between the dependent variable y and the independent variable x can be represented as a straight line in a two-dimensional plane.

LASSO

Least absolute shrinkage and selection operator (LASSO) is a regression analysis technique that performs both features regularization and subset selection over the available features, improving the accuracy of the prediction while providing more interpretable models. Before LASSO, stepwise selection was used to select a subset of the covariates of the model, but since it is a discrete process (a feature can either be included in the model or not) small changes in the data can highly affect the quality of the final model, sometimes reducing the accuracy. For feature regularization, the prevalent technique used to be ridge regression, which regulates large coefficients to reduce overfitting, yet it does not address the interpretability of the model. LASSO combines the two approaches and preserves their advantageous characteristic by shrinking some coefficients and setting others to zero.

The optimization objective for LASSO is:

$$\min_{\beta} \frac{1}{2n_{samples}} \|X\beta - y\|_2^2 + \alpha\|\beta\|_1 \tag{5.4}$$

Where α is a hyperparameter of the model that controls the degree of sparsity of the estimated coefficients. With $\alpha = 0$, we have again an OLS linear regression model.
The implementation in scikit-learn uses coordinate descent as the algorithm to fit the coefficients.

Support Vector Machines

An SVM model constructs one or more separators (hyperplanes) as a decision surface such that the distance of datapoints of different classes is maximized. When this approach is applied to a regression problem, it is called Support Vector Regression (SVR). The goal of SVR is to compute y_i from the observation x_i with a margin of tolerance ϵ. The objective function is the following:

$$\mathbf{y} = \beta\mathbf{X} + b, \qquad \min J(\beta) \tag{5.5}$$

$$J(\beta) = \frac{1}{2}\|\beta\| + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right) \tag{5.6}$$

Where ξ_i and ξ_i^* indicate the distance of datapoints that lie outside the margin ϵ, in both sides of the hyperplane, respectively. C is a hyperparameter regulating the penalty for errors. The optimization problem is computationally simpler to solve in its Lagrangian dual formulation. The dual formula is derived from the primal by introducing non-negative multipliers α_n and α_n^* for each observation x_n, and it is expressed as:

$$L(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(a_i - a_i^*)(a_j - a_j^*)\langle x_i^\top x_j\rangle + \epsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) \quad (5.7)$$

The parameter β can be described as a linear combination of the training observations with the following equation:

$$\beta = \sum_{n=1}^{N}(\alpha_n - \alpha_n^*)(x_n^\top x) + b \quad (5.8)$$

Some regression problems cannot be accurately described by a linear model. The dual formulation allows to extend the approach to non-linear functions by substituting the dot-product $\langle x_i^\top x_j\rangle$ in the formula with a nonlinear kernel function $\mathbf{G}(x_i, x_j) = \langle\phi(x_i)\phi(x_j)\rangle$. $\phi(x)$ is a transformation that maps x to a high-dimensional space. The classifier is again a hyperplane in the transformed space, though it may not be linear in the original space. Table 5.4 reports three different kernels that we explored in this work, while Figure 5.5 graphically highlights the difference between SVM models trained with a linear, polynomial and rbf kernel for a toy regression example. In our approach we experimented and validated empirically the use of the rbf kernel.

Kernel name	**Kernel function**
Linear	$\mathbf{G}(x_j, x_k) = x_j^\top x_k$
Polynomial	$\mathbf{G}(x_j, x_k) = (x_j^\top x_k + 1)^n, \quad n = \{2,3,...\}$
Radial basis function (rbf)	$\mathbf{G}(x_j, x_k) = \exp(-\gamma\|x_j - x_k\|^2), \quad \gamma > 0$

Table 5.4. Common SVM kernels. γ, α and c are parameters of the functions that need to be tuned for the application. γ is often set to $1/2\sigma^2$.

5.3.4 Model Optimization

We also refer to this phase as *Tuning* of the algorithms. SVM and LASSO, in fact, require to set some parameters before running the training phase, and the choice of their values is not a trivial task. The domain of those parameters can be very large and continuous, often $(-\infty, +\infty)$, and different values for the same parameter may result in surprisingly different models and prediction results. For this reason, this is a phase that requires particular attention, and a lot of time for research and assessment of the best parameters values, which is usually done by testing numerous configurations and gauging their performances for a certain task.

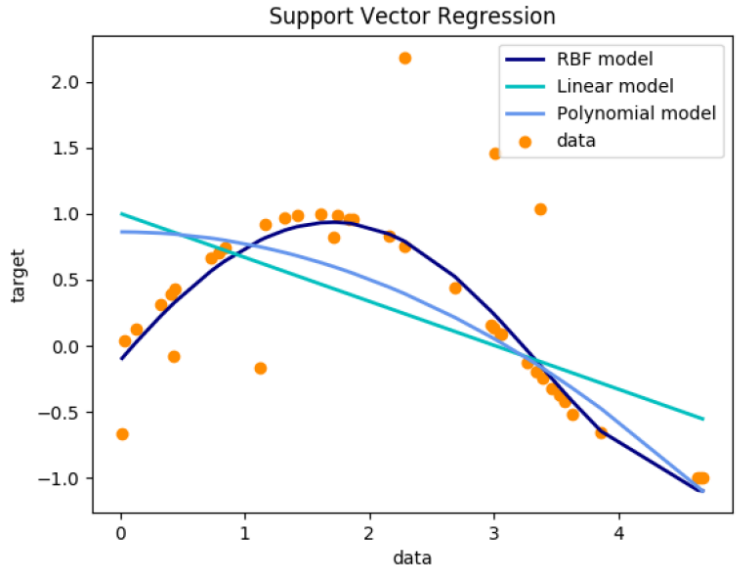

Figure 5.5. Accuracy of SVM models employing a linear, polynomial, and rbf kernel. As we could expect, linear kernel perform the worst while rbf has the highest accuracy, correctly modeling all datapoints, excluding outliers. From scikit-learn documentation.

There is no such thing as a perfect configuration that works every time, because for each specific problem there can be one that best suits the context and the data.

We need to properly tune SVM and LASSO algorithm to ensure that the models are trained using the best configuration for our problem. For SVM, we consider the following hyperparameters:

Kernel: linear, polynomial, radial basis function, hyperbolic tangent. It highly depends on the datapoints distribution in the feature space, as it is shown in Figure 5.6.

C: the penalty factor that regulates the trade-off between misclassification and simplicity of the decision surface.

Gamma: indicates how far the influence of a single training example reaches, with low values meaning far and high values meaning close.

Degree: of the polynomial equation, only meaningful for polynomial kernel.

While for LASSO we only consider one parameter, α, that regulates the trade-off between the minimization of the residual sum of squares and minimization of

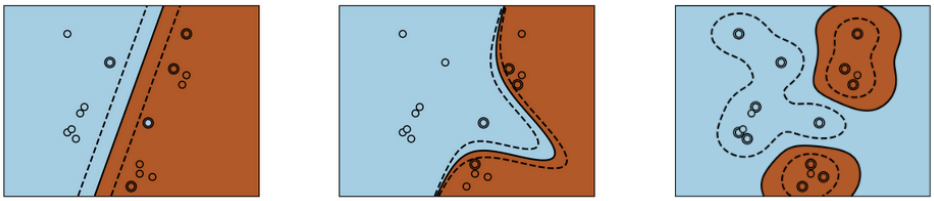

Figure 5.6. Comparison of the behavior of three different SVM kernels for classification. From left to right, linear, polynomial and rbf. Linear kernel is the simplest one, yet it represents a good option when datapoints are linearly separable. Rbf is capable of tracing complex discriminative boundaries with various shapes. From scikit-learn documentation.

the sum of squares of the coefficients. More practically, α controls the size of the selected subset of features and the entity of their shrinkage. High values result in the selection of a smaller subset and in greater shrinkage. Setting $\alpha = 0$ brings LASSO back to classic linear regression. Figure 5.7 clearly shows the effect of different values of α on the shrinkage of the coefficients.
In Chapter 6 we report the different configurations we tested together with those that performed the best on the training data.

To evaluate the goodness of the numerous configuration, we need a *Loss Function*. A loss function maps events or values of a set of variables onto a real number, which represents the cost of the event. An optimization problem, as is ours, aims to minimize the loss function, meaning that it is desirable to reduce the cost, or loss, of an event as much as possible. In our case, the event is the prediction of personality trait scores, and the set of variables are the predicted and actual values. The difference between those values is the loss of the model, which we want to minimize. The loss function we choose to adopt is Mean Squared Error (MSE), which is the mean squared difference between predicted and actual values

$$MSE(P, A) = \frac{1}{n}\sum_{i=0}^{n-1}(p_i - a_i)^2 \tag{5.9}$$

Where with $P = (p_1, p_2, ...p_n)$ are predicted values and $A = (a_1, a_2, ..., a_n)$ are actual values of the training samples that we are evaluating the model on. MSE is a positive score in the range $[0, +\infty)$, and smaller values indicate better accuracy.

The purpose of model optimization is not only selecting the configuration that best fits the data, but also to determine the machine learning algorithm that has the highest performance in this particular context. We observe this algorithm to be SVM, and we discard Linear and LASSO regression. In Chapter 6 we report the results of the optimization in terms of mean squared error.

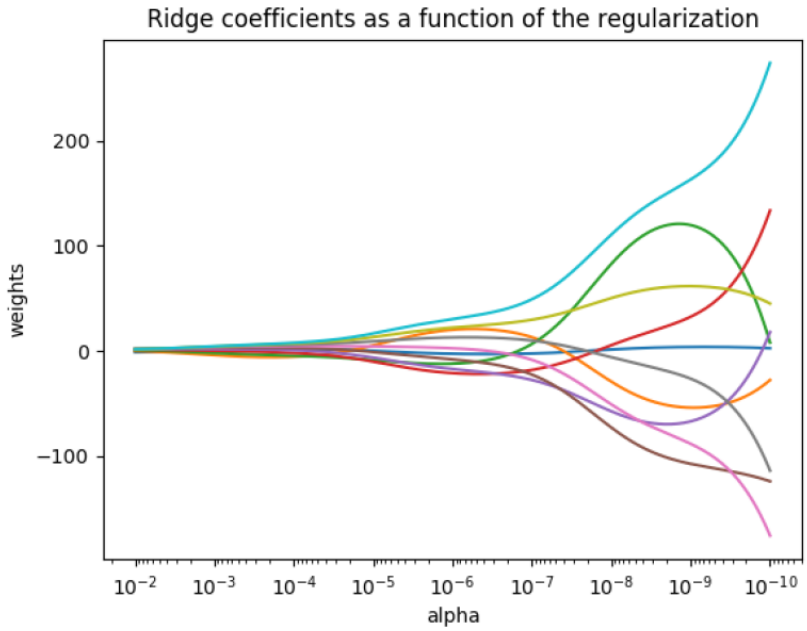

Figure 5.7. Effect of the regularization parameter on the coefficients of a ridge regression estimator. High values cause all coefficients to be close to zero, whereas with very low α all coefficients exhibit big oscillations. From scikit-learn documentation.

Unless differently specified, all the models described in the following sections are trained using SVM.

5.4 Twitter API

To interact with Twitter, both for authentication and sending API requests, we use the open source Python library *Tweepy*. Tweepy is a simple and lightweight library that provides helpful functions for accessing the Twitter API. To access those functions, similarly to other Social Media Sites (e.g. Facebook, Instagram) it is required to create an application through the platform provided by Twitter[4]. We named our application *TwitPersonality*, whose basic information are showed in Figure 5.8.

[4] https://apps.twitter.com/

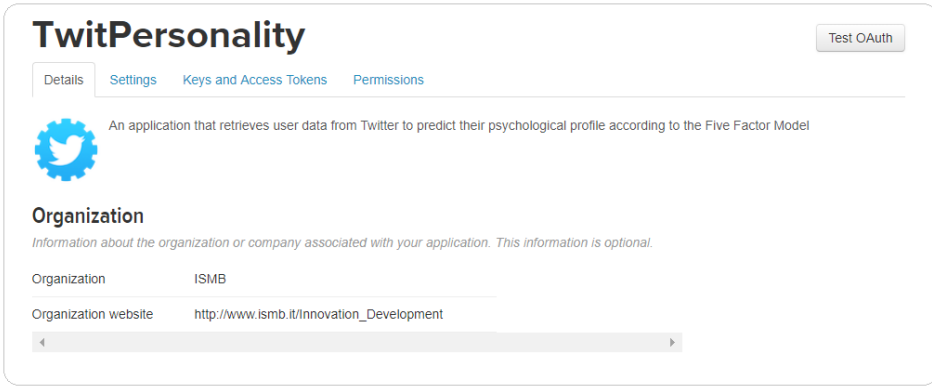

Figure 5.8. Summary information about TwitPersonality application, as they are presented in the Twitter Apps Management page.

Twitter apps act like gateways between Twitter content and what instead lies outside of it, and they need authentication. Twitter uses OAuth to provide authorized access to its API, which is a standard used to grant websites or applications the access to other websites without revealing the password. In our case, we need Twitter to grant our application the access to the API functionalities.
Oauth requires developers to be in possess of a valid access token. An access token controls what API endpoint can be queried, and so what actions can be carried out using the application. In practice, it is simply a long string of letters and numbers. the authentication process is defined by the following steps:

1. Get a request token from Twitter

2. Redirect the user to twitter.com to authorize the application. A message like the one in Figure 5.9 will be showed.

3. If using a callback, Twitter will redirect the user to us. Otherwise the user must manually supply us with the verifier code. Since we are developing a desktop application, we cannot use callbacks, so we must manually input the verifier code. It is simply a GET query parameter in the URL of the page that will load after authorizing the app.

4. Exchange the authorized request token for an access token.

Currently, Twitter access tokens do not expire, so it is possible to save it locally for later use, avoiding the need to repeat all the authentication steps every time we use the application.

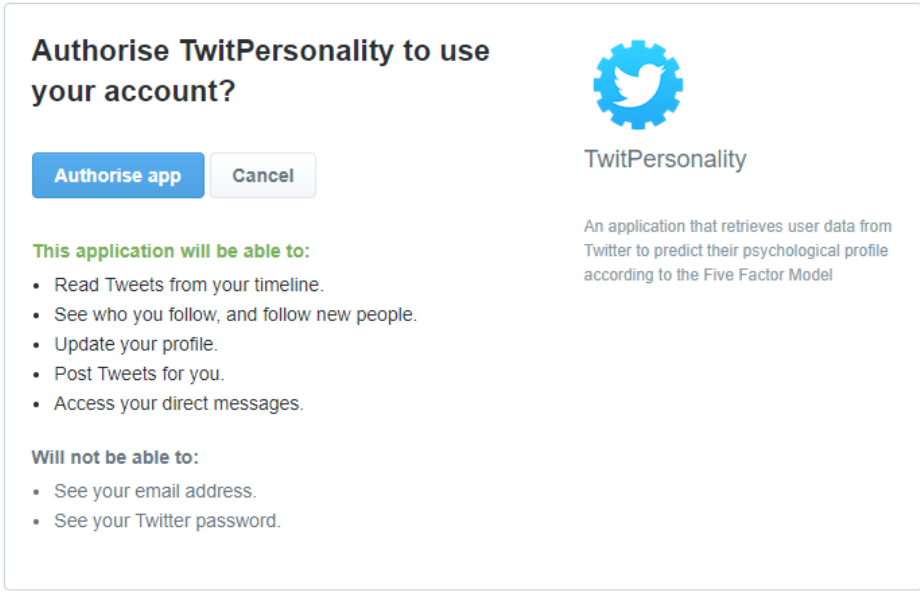

Figure 5.9. Application authorization message. It is clearly shown what permissions the app is requesting. In our case, we only need the first two.

After successfully obtaining and access token, we are authorized to perform all the actions reported in the list of Figure 5.9, though we are only interested in reading the tweets and the number of followers, the latter for statistic purposes. For the list of tweets in the timeline, we send a GET request to the API endpoint *statuses/user_timeline* where we need to specify the username that we are interested in and the number of tweets we want to retrieve with a single call. The final http request will look like:

```
GET https://api.twitter.com/1.1/statuses/user_timeline.json
?screen_name=<username>&count=2
```

Twitter allows to get up to 200 tweets with a single request, and paginates the responses, meaning that they contain a pointer to the next chunk of tweets. To retrieve them, we simply send a GET request to that pointer. However, Twitter let us download at most 3,200 tweets for a given user. This is not a problem, since all users in our experiment feature less than 3,200 tweets, and even if someone exceeded that limit, we still would have enough information for predicting personality.

Regarding the number of followers, this information is included in the user's basic information, accessible at *users/show* with the http request

```
GET https://api.twitter.com/1.1/users/show.json
?screen_name=<username>
```

this call returns a wide range of information, including name, Twitter id, language, location, number of tweets, number of followers and following, and others. Since we are only interested in the number of followers, we ignore other fields.

Table 5.5 sums up the properties and characteristics of the endpoints and relative requests.

	Tweets	**Followers**
API endpoint	statuses/user_timeline	users/show
HTTP request	GET	GET
Parameters	screen_name, count	screen_name
Returns	json	json
Rate limit/15 mins	1500	900

Table 5.5. Summary table of the API endpoints we need and their basic characteristics. Although Twitter limits the number of API calls that can be made on a given endpoint, those limits are rather high and we never exceed them during our research.

5.5 Predicting Personality from Tweets

Our approach is based on transfer learning: we learn a model on a set of status updates from Facebook with the aim of testing that model on users' tweets from Twitter. To test our model we collect a test set of Twitter users who gave their consensus on analyzing their profiles and at the same time answered a psychological questionnaire to measure their personality trait scores.

The process described below is relative to a single Twitter user. For many of them, it is simply repeated thoroughly.

Given a Twitter handle (e.g. @username), we crawl the Twitter API to retrieve all available tweets of that user. We have already mentioned previously that tweets are somewhat different from status updates, despite being generated in similar Social Networks and having similar characteristics. For this reason, we define some additional preprocessing steps that are specific for tweets and are executed before those listed in Section 5.3.1.

In detail, they are:

Pure retweet removal: When retweeting a tweet (either own of someone else's), it is possible to add some text along with it. In that case, it will appear as a normal textual tweet followed by the URL to the retweeted one. Instead, if no personal text content is added, it will be represented as *RT @tweet-owner: tweet-text tweet-url*. We refer to this case as pure retweet. Since we base the prediction only on the textual features created by a specific user, all retweets without additional content are removed, because they either belong to a different individual, or will be retrieved twice.

URL removal: URLs may appear in retweets, content sharing from external sources (e.g. YouTube videos), or other cases. In the context of our study they are not informative, so they are removed from the tweets.

Mentions removal: Users often mention their friends in their tweets, adding some noise to the text data; mentions are also found in retweets. We remove all mentions from our data.

Hashtags removal: We choose not to remove the whole word, as in the previous cases. Instead, we keep the original word without the hash (#) symbol. This is because some tags consist of meaningful words (e.g. "I love #star #wars") which we do not want to lose. This solution effectively addresses also the case of non-existing hashtag words (e.g. "#netNeutarlity2017") that are ignored in the subsequent steps, when a corresponding word embedding will not be found.

All the processing steps detailed above are carried out using regular expressions, with the exception of hashtags removal. In fact, hash symbols are automatically deleted by punctuation removal. Apart from those further refinements, we process tweets the same way as status updates in order to derive a feature vector of 900 dimensions. Figure 5.10 illustrates the transfer learning context highlighting the two different domains for training the model and testing it.

At this point we have all the tweets of the user correctly preprocessed and transformed in the corresponding representation in the 900-dimensional vector space. The last step is feeding those vectors to the Big Five models to get a personality score of the individual.

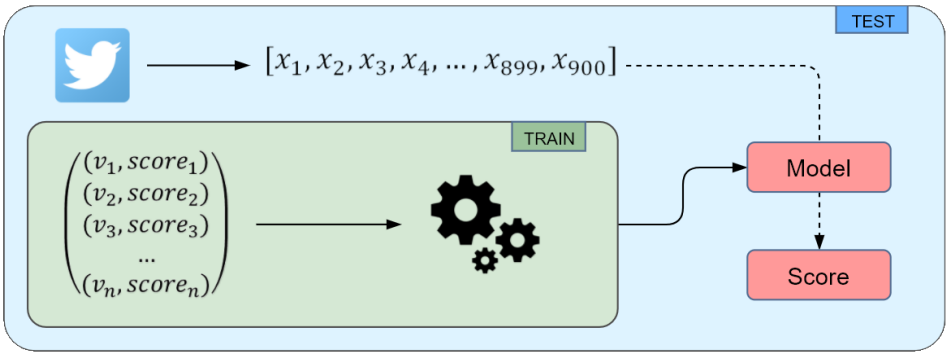

Figure 5.10. Each pair of transformed status updates and corresponding personality trait score is a training sample that we feed to the algorithm. We train five different models, one for each personality trait, and test them with tweets. After applying tweet-specific processing steps, a tweet is transformed into a vector the same way as status updates.

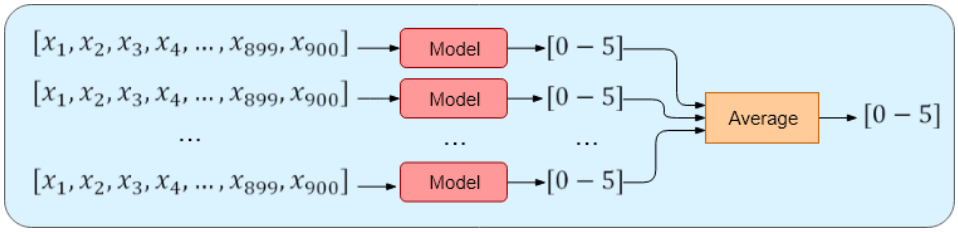

Figure 5.11. Prediction of a personality trait score. Each tweet is individually fed to the model that computes a value. All values are averaged to derive the personality trait score. The process show in figure is repeated for each Big Five trait.

Generally, users share more than one tweet in their personal profile, so we need to devise a strategy to obtain a single personality score from all of them. To replicate as closely as possible the same scenario of the training phase, in which each status update is individually fed to the algorithm, we separately feed each tweet to the trained models. This way, we obtain a number of scores for each trait that is equal to the number of tweets. The final score is computed as the mean of all the predictions. This process is summed up and made more clear in Figure 5.11.

We test our approach on the Twitter user sample that we gathered, and results are reported in Chapter 6.

Chapter 6

Experimental Results

We first evaluate our approach on the myPersonality dataset comparing the performances of the three algorithms introduced in Chapter 5; we then verify the transfer learning capability of the model with an in-house built gold standard created from twenty four panelists.

6.1 Technical Details

We use *Python* to develop all the parts of the application. In particular, we use *Scikit-learn* [129] to achieve most of the operations, such as training and testing the models, saving the trained models on disk, computing mean squared error or preprocessing status updates and tweets. Moreover, we use *Numpy* and *Pandas* to efficiently manage data structures. Finally, we use *Tweepy* to download users' tweets from the Twitter API platform.

Each separate step of the whole process is carried out by a single Python script, which is unrelated to the others. For example, we use different scripts to optimize and actually learn the predictive models and store them on disk. Similarly, tweets download and personality predictions using them are run by other two scripts. We also aggregate a number of useful functions for reading and parsing dataset files into two respective scripts. We choose to keep each step separate from the others mainly for simplicity (each script is responsible only for a small part of the application) and for testing (so we do not need to test the whole application if we are only interested in a part of it). Besides, it is possible to use a bash script to run the whole Python pipeline with a single command.

The machine running the scripts features a quad-core Intel core i7 6700HQ @2.60GHz and 16GB or RAM.

6.2 Model verification

To derive the best performing predictive model we explore different machine learning algorithms and configurations. We evaluate the models on the training set with the objective of minimizing the mean squared error of the predicted personality scores, which is our loss function.

For both SVM and LASSO we carry out a tuning phase to select the best configuration of the hyperparameters for our application. This is a crucial phase since minor differences in the configurations of the algorithms' parameters may lead to significantly different results. We have introduced and briefly described the hyperparameters in Section 5.3.4. C, γ and α have continuous values in $\mathbb{R}$, whereas *degree* is a positive number in $\mathbb{N}$, hence the available configurations are infinite. We explore a subset of them, for a total of thirty different models to be tested. Parameters that we considered, along with their different values, are reported in Table 6.1.

Algorithm	**Parameter**	**Value**
SVM	Kernel	*linear*, *rbf*, *poly*
	C	1, 10, 100
	Gamma	0.01, 0.1, 1, 10
	Degree	2, 3
LASSO	Alpha	1^{-15},1^{-10},1^{-8},1^{-5},1^{-4},1^{-3},1^{-2},1,5,10

Table 6.1. Hyperparameters and their values for SVM and LASSO models. For SVM, rather than using an exhaustive grid search approach, we only test a subset of all the possible combinations, for a total of 19.

We observe that SVM performs better than other algorithms. For each personality trait exists a SVM configuration with a minimum MSE lower than that of other learning models. Table 6.2 reports those configurations with relative error value, that is computed using 10-fold cross validation on the training set. All the best performing models use a rbf kernel and similar values for C and *gamma*. Openness presents the lowest error while Extraversion has the highest. Optimal configuration for linear and LASSO regression are reported in Table 6.3. We can also observe that LASSO performs slightly worse than SVM, and all the best performing models have the same value for α, while linear regression, as expected, has the highest MSE values overall, and thus performs the worst. Table 6.4 sums up the mean squared error of the different models and highlights the margin of improvement of SVM over other algorithms.

Trait	Kernel	C	Gamma	MSE
Openness	rbf	1	1	0.3316
Conscientiousness	rbf	10	1	0.5300
Extraversion	rbf	10	1	0.7084
Agreeableness	rbf	10	1	0.4477
Neuroticism	rbf	10	10	0.5572

Table 6.2. Best-performing SVM configurations of kernel and hyperparameter values used in our approach segmented per trait and measured by mean squared error.

Trait	Model	Configuration	MSE
Openness	LReg	-	0.3915
	LASSO	alpha = 0.0001	0.3345
Conscientiousness	LReg	-	0.6200
	LASSO	alpha = 0.0001	0.5363
Extraversion	LReg	-	0.8210
	LASSO	alpha = 0.0001	0.7106
Agreeableness	LReg	-	0.5407
	LASSO	alpha = 0.0001	0.4500
Neuroticism	LReg	-	0.6625
	LASSO	alpha = 0.0001	0.5595

Table 6.3. Mean squared error values for Linear Regression (LReg) and LASSO regression for the five factors. In LASSO, the best-performing configuration always results for $\alpha = 0.0001$.

As we mentioned in Section 5.3.2, we choose to use concatenation for representing tweets and status updates in the vector space. We motivate this choice with experimental results in Table 6.5, which shows an average value and a standard deviation of MSE across all nineteen tested SVM configurations[1] and five personality traits. We note that although other transformation methods sometimes achieve a

[1]Configurations: linear kernel with C=1; poly kernel with all combinations without repetition of degree=[2,3] and C=[1,10,100]; rbf kernel with all combinations without repetition of γ=[0.01, 0.1, 1, 10] and C=[1, 10, 100].

Model	OPE	CON	EXT	AGR	NEU
SVM	0.3316	0.5300	0.7084	0.4477	0.5572
LASSO	0.3345 (0.0029)	0.5363 (0.0063)	0.7106 (0.0022)	0.4500 (0.0023)	0.5595 (0.0023)
LReg	0.3915 (0.0599)	0.6200 (0.0900)	0.8210 (0.1126)	0.5407 (0.0930)	0.6625 (0.1053)

Table 6.4. Summary of mean squared error values for all models. LASSO performs almost as well as SVM, while linear regression (LReg) has a more consistent margin of error. Those differences are somewhat consistent across all personality traits. Both for Linear Regression and LASSO we also report the increment in terms of mean squared error with respect to SVM.

lower MSE, standard deviation for concatenation is the lowest for each personality trait, suggesting that this method is more stable and consistent, hence more reliable.

Method	OPE		CON		EXT		AGR		NEU	
	mean	std	mean	std	mean	std	mean	std	mean	std
sum	0.459	0.308	0.723	0.542	0.939	0.655	0.6	0.429	0.75	0.497
maximum	0.352	0.014	0.547	0.023	0.735	0.031	0.461	0.018	0.58	0.028
minimum	0.352	0.016	0.546	0.024	0.732	0.032	0.462	0.017	0.579	0.026
average	0.355	0.015	0.547	0.025	0.735	0.03	0.464	0.019	0.582	0.034
concatenation	0.352	0.01	0.548	0.016	0.736	0.026	0.463	0.018	0.583	0.024

Table 6.5. Average MSE value and standard deviation of the nineteen SVM models across all the different configurations that we take into consideration for tuning the hyperparameters. Mean values, except for sum, are almost equivalent, though standard deviation for concatenation is the lowest for all the five factors.

We test our models using pre-trained word embeddings. We explore different datasets and evaluate how they affect the predictive power of the models in terms of mean squared error. FastText is an open source library from Facebook AI research (FAIR), and it has been already introduced in Section 5.2. Li et al. [113] explored different sources and text processing combinations to train word embeddings. They learned word vectors both from tweets and from general text data available on the Web, including "spam" tweets (less informative ones), and considered both words and words plus phrases for the models. The combination of those approaches led to ten different datasets. Although learning embeddings from tweets should improve the overall performance of the model, this highly depends on the context of the application and on the data it is tested on, and we observe that in our case the model from FastText performs better. Table 6.6 shows some statistics of the datasets we explored. We notice that despite being trained on 2.8 times more words on average, those datasets are able to achieve only up to 3.14% (π) more

word coverage than FastText.

Dataset	word coverage	# of word vectors	avg OCEAN MSE
FastText	95.08%	1M	0.517
[113] Dataset 1	96.02%	1.9M	0.532
[113] Dataset 2	95.31%	2.9M	0.551
[113] Dataset 3	96.36%	2.7M	0.558
[113] Dataset 4	95.69%	4M	0.541
[113] Dataset 5	96.49%	1.4M	0.539
[113] Dataset 6	95.91%	3.1M	0.518
[113] Dataset 7	98.05%	1.7M	0.52
[113] Dataset 8	97.45%	3.7M	0.541
[113] Dataset 9	98.22%	2.2M	0.533
[113] Dataset 10	97.65%	4.4M	0.537

Table 6.6. Statistics of the pre-trained word embeddings models we use in our experiments. Although some of the other datasets achieve lower MSE for one or more personality traits, the mean value across all traits is still higher than the one relative to FastText. Mean squared error is computed on the models trained with configuration reported in Table 6.2. Datasets 1-10 differ in terms of text source (general data vs tweets), tweets filtering (spam tweets included or not) and text processing (words vs words and phrases).

6.3 MyPersonality Big

All the steps described in Section 6.2 are carried out using as training set the gold standard from myPersonality detailed in Section 5.1. It contains 9,913 status updates of 250 Facebook users, which are lowered to roughly 8,700 after removing those that are too short. The size of the actual training set is therefore quite modest, yet the results achieved are promising. We refer to this dataset as myPersonality small.

MyPersonality small is just a sample of the entire dataset, which stores information about all the users who participated in MyPersonality experiment during the time period the Facebook application was active, and is available upon formal request to the team of researchers who collected it. We aim to extend the analysis of Section 6.2 to the whole myPersonality dataset, in order to test the predictive power of the models trained on significantly more data. We do not repeat the

model optimization step as we only extend to a bigger dataset the results obtained on a sample of it.

Differently from the case of myPersonality small, in which data is stored in a single file, in this case there are several files available, each one containing a specific piece of information about the users of the project. This is most likely due to the huge size of those files. For our analysis, we are only interested in two of them, the first storing status updates and the other storing Big Five personality scores. In both files it is also reported the user id of the person whose the status updates and the personality scores belong to, so it is possible to merge the two files on the user id, creating a new one similar to the dataset of myPersonality small. We will refer to this file as myPersonality big.
Due to their huge size in terms of number of lines, to efficiently perform the merge operation we load the two files into *DataFrames* and compute the inner join between them. A DataFrame is a two-dimensional labeled data structure provided by the Python library *pandas*, whose columns may be of different types. A large number of operations can be performed on dataframes and this is particularly useful for our purpose. The result of the join operation is a new DataFrame which stores a list of records of the type

```
user id, status update, Ope, Con, Ext, Agr, Neu
```

featuring only the user ids that appear in both the original files.

MyPersonality big stores a total of 16,500,000 records, corresponding to as many status updates from 116,000 distinct users. Such great numbers allow to perform different types of analyses that were not possible before due to the limited size of the dataset. Ironically, we now have much more data than we need. In fact, training a SVM model on a training set in the order of millions of records is rather unfeasible for the machine we use (see Section 6.1), not to mention that it may lead to overfitting, which we want to avoid.
For these two reasons, we sample myPersonality big and train the SVM models subsets of variable dimensions, depending on the particular approach that we want to test. We execute three different analyses, which differ from one another in terms of how we treat the train and test split of each cross-validation iteration. We implement 4-fold cross-validation in all the approaches, and test the three of them with different subsets of the training set. The techniques explained below are repeated five times, once for each personality trait.

Test 1

It is the simplest one, and the same as the one of myPersonality small. Although we have already trained a model using this approach, we are interested in comparing

the two datasets, small and big, using similar data and techniques.
We feed to the machine learning algorithm a number of training samples in the form (*status update*, *Big Five scores*). We subsequently test the trained model using the remaining test samples. Mean squared error is computed by averaging the errors for each test sample, and is reported in Table 6.7 for each personality trait and data subset. We note that as the size of the training set increases the average MSE value gets lower. However, except for Conscientiousness and Extraversion, values are higher than those achieved on myPersonality small (Table 6.2).

Status Updates	OPE	CON	EXT	AGR	NEU
5000	0.4284	0.5196	0.7115	0.4986	0.6524
7500	0.4261	0.5179	0.7189	0.4865	0.6453
10000	0.4184	0.5101	0.6971	0.4799	0.6459
12500	0.4189	0.5086	0.6975	0.4768	0.6484
15000	0.4228	0.5147	0.6911	0.4759	0.6459
17500	0.4216	0.5136	0.6874	0.4755	0.6517
20000	0.4181	0.5066	0.6816	0.4773	0.6444

Table 6.7. Mean squared error values for test 1, in which training and test samples are individually fed to the algorithm.

Test 2

Before feeding training data to the algorithm, we aggregate records per user, this way we keep track of which user wrote a certain status update. Then, we compute the cross-validation split on the number of users rather than on the number of total records. This is based on the assumption that status updates are somewhat equally distributed across all the users, so splitting users has a similar effect to splitting records. The test seems to prove this hypothesis, as the ratio between number of status updates and number of users for the various subsets is constant in a small range.
We train the algorithm the same way as test 1, but the MSE is computed for a single user on all his status updates, then it is averaged with that of all other users. The reason we test this technique is because we want to assess the predictive power of the model on a single user, since this will be the real scenario in which the model will be applied. Test results are reported in Table 6.8. For some traits, training on 50 users produces lower MSE values than bigger subsets. This may depend on the small size of the dataset, which causes the model to not converge properly and compute unusual scores.

Users	OPE	CON	EXT	AGR	NEU
50	0,5455	0,7015	0,714	0,5521	0,5837
100	0,4667	0,6364	0,7678	0,428	0,7128
150	0,463	0,6326	0,7443	0,4888	0,7264
200	0,4655	0,616	0,7451	0,4626	0,675

Table 6.8. Mean squared error values for test 2. The four different subsets of users roughly correspond, respectively, to 6000, 10000, 15500, and 21000 status updates.

Test 3

The whole analysis is at user-level. We analyze the entire dataset to extract data in the following form: [*user id*, *list of status updates*, *Big Five scores*]. Then, for each user, we average all vector space representations of his status updates into a single one; the average operation is performed the same way as the one in 5.3.2. This results in a number of records equal to the distinct users in the dataset, which are about 116,000. It is still not feasible for our machine to train the SVM model on the whole training set, so we try various subsets. Training and test samples are again the form (*status updates*, *Big Five scores*), as in Test 1, with the difference that the 900-dimensional vector now represents all the status updates of a certain user, rather than just one of them. We again compute MSE by averaging the errors of all the test samples.

However, we choose not to further pursue this approach, because we believe that averaging all the status updates of a user into a single one we may lose semantic and emotional information by squeezing them into their median value. Although on the one hand this way we smooth out biases and uncommon personality manifestations, on the other hand the resultant vectors are less informative and provide a poor informative contribution, thus lowering the discriminating power of the model. Table 6.9 reports test results. We note that except for Agreeableness, mean squared error values are lower than the ones of previous tests. However, there is not an appreciable decrease of MSE as the number of status updates grows, and sometimes we can observe an opposite trend. This behavior demonstrates that our concern about averaging all the status updates into a single one is well-founded.

Although Test 1 and Test 2 differ one another in terms of how the MSE is computed on the test set, the models are trained in the same way, namely, by feeding couples of (*status update*, *Big Five score*) to the algorithm. Since Test 2 shows promising MSE results and seem to converge, we choose to adopt this approach to train the predictive models on myPersonality big. In this case, we do not split the data in train and test set, but we use all of it to train the models,

Status Updates	OPE	CON	EXT	AGR	NEU
5000	0.4284	0.5196	0.7115	0.4986	0.6524
7500	0.4261	0.5179	0.7189	0.4865	0.6453
10000	0.4184	0.5101	0.6971	0.4799	0.6459
12500	0.4189	0.5086	0.6975	0.4768	0.6484
15000	0.4228	0.5147	0.6911	0.4759	0.6459
17500	0.4216	0.5136	0.6874	0.4755	0.6517
20000	0.4181	0.5066	0.6816	0.4773	0.6444

Table 6.9. My caption

which will be tested on real users from Twitter.

6.4 Transfer learning assessment

In order to empirically validate our transfer learning approach, we created a gold standard of Twitter users, posts, and personality trait scores. In the remainder of this paper, we refer to this gold standard as Twitter sample. We asked twenty four panelists to provide their Twitter handle and to complete the Big Five Inventory (BFI) [127] personality test.

Twenty six panelists took part in the creation of the Twitter sample, of which two were discarded because of the insufficient number of available tweets. Table 6.10 reports some statistics about the Twitter sample and Figure 6.1 shows the histograms of the personality scores obtained from the survey of the five personality traits.

Statistic	Value	Lowest	Average	Highest
Total users	24	-	-	-
Total tweets	18,473	-	-	-
Tweets per user	-	9	769.7	2,252
Avg words per tweet per user	-	5	6.8	8.8
Number of followers	-	12	1,375.5	20800

Table 6.10. Basic statistics about the Twitter user sample. All the values reported are calculated after having applied all the preprocessing steps.

We evaluate the predictive power of our approach using the best performing

	OPE	CON	EXT	AGR	NEU
max	4.8	4.78	4.38	4.33	3.63
min	2.5	2.33	1.75	2.78	1.5
avg	3.8917	3.5513	3.2208	3.6438	2.6642
std	0.5763	0.5682	0.5449	0.3707	0.5250

Table 6.11. Statistics about the personality traits of the Twitter user sample. We report standard deviation, highest, lowest and average value across all twenty four users. Where OPE=Openness, CON=Conscientiousness, EXT=Extraversion, AGR=Agreeableness, NEU=Neuroticism.

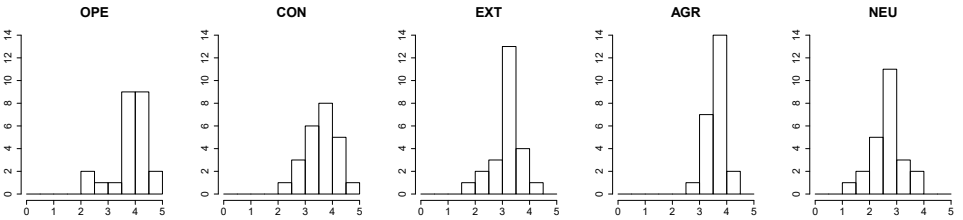

Figure 6.1. Five histograms representing the distributions of the Big5 personality traits of the Twitter sample.

models assessed with the myPersonality datasets over the gold standard of Twitter users. We download and process the tweets as described in Section 5.4, and consider each of them as a single test sample that is then fed to the SVM models, whose predicted score is averaged to compute the final trait score (see Figure 5.11), this is repeated for all the five personality traits. A different approach that could be followed consists in averaging all the tweet vectors before feeding them to the model. However, we discarded this option for the same reason explained in Section 6.3: by averaging all the tweets into a single one, all the emotional and syntactic features that are expressed in them are smoothed out in favour of their median value, and the model can leverage less information to compute the predictions.

In Table 6.12, we sum up mean squared error values obtained in the tests described in the previous sections. We report MSE obtained on the training set as the average of all cross-validation iterations (10-fold for myPersonality small and 4-fold for myPersonality big), and MSE of the actual (expressed by the panelists though the survey) and predicted values (computed by our approach) with the predictive models trained on both the datasets. We note that our approach achieves significantly better results when applied to the Twitter sample and its newly created in-house gold-standard of personality scores. Except for Openness in the experiment with myPersonality small, our approach achieves lower MSE scores on all

traits. In addition, for this particular Twitter sample, models trained on myPersonality small outperform those trained on myPersonality big. However, the modest size of the Twitter sample cannot account for enough variation in the personality traits, whose distribution is rather biased, and this may affect the results of the predictions; we will further investigate this aspect in future work.

	OPE	**CON**	**EXT**	**AGR**	**NEU**
myPersonality small	0.3316	0.5300	0.7084	0.4477	0.5572
Twitter sample (MP small)	0.3812	0.3129	0.3002	0.1319	0.2673
myPersonality big	0,4655	0,6160	0,7451	0,4626	0,6750
Twitter sample (MP big)	0.3178	0.3236	0.4110	0.1362	0.2803

Table 6.12. Summary table reporting mean squared error computed on cross-validation and on the Twitter sample, for both the myPersonality small (MP small) and big (MP big) datasets. In both cases the models are trained with the configurations of Table 6.2. Results reported for myPersonality big refer to Test 2.

Chapter 7

Conclusions

We presented a supervised learning approach to compute personality traits by only relying on what an individual publicly tweets about his thoughts. The approach segments tweets in tokens, it does ad-hoc preprocessing of tweets, then it learns word vector representations as embeddings that are then combined and used in a supervised learner SVM classifier. Our approach has developed five different learning models, one for each personality trait. We tested the convergence of our approach using an international benchmark of Facebook status updates in a controlled experiment and we observed low mean squared errors from the predicted vs the actual values. We then applied the learned models with a Twitter sample that we generated in-house. The generation protocol of the sample followed a well-defined psychological test and we collected actual values of personality traits from twenty four panelists who gave their consent in collecting and processing their tweets. We then tested our five learning models and observed lower MSE values than those obtained with the Facebook international benchmark dataset.

We used pre-trained word embeddings from FastText. We aim to compute word representations on textual data extracted from Twitter and other social networks. This way, we expect to obtain a higher word coverage and to learn more meaningful word embeddings, that are able to capture semantic and syntactic features that are more typical of social networks. In addition, we plan to implement the tweet transformation phase using a convolutional neural network (CNN) to aggregate words and n-grams into a single vector representing the tweet, which can then be used to obtain a personality score for the users. At present, we transform each tweet into a sequence of 900 real-valued numbers derived from the embeddings of the words used in the tweet itself. The number of features could be expanded by incorporating additional semantic knowledge, such as named entities or sentiment information. The current approach has been tested on English content only, we

also plan to extend it to other languages exploiting the similarity of word meanings in the vector space. Finally, we aim to extend the Twitter sample by acquiring more panelists running our questionnaire.

It is arguable that the virtual identity that is presented on social networks is a reflection of the true personality of individuals or it is merely a self-idealization of it. If the second hypothesis was true, personality scores predicted leveraging social network features would not be accurate and reliable. However, research seem to prove it wrong [128]. In addition, language features, including word choice, which is what we use to derive personality, should be less affected by this issue.

Direct uses of this work include personalization services, recommender systems and marketing applications. For example, a travel agency could employ a commercial software which suggests specific destinations based on the personality of its customers. The suggestion could be further refined by also selecting the characteristics of the hotel and providing some activities to carry out during the holiday. Streaming services such as Spotify and Netflix can recommend new songs to listen to or new movies or tv-shows to watch, depending on the preferences derived from the particular personality of a user. Such a feature could enhance the already existing recommender system that is used by both the streaming services, and it is particularly easy to implement since they both allow users to log in using their social networks accounts.

Just in these days, a big scandal has involved Facebook and Cambridge Analytica, a company which operates in the field of data mining and data analysis. The company has allegedly collected a huge amount of data from Facebook users without their consent and knowledge, and used it to predict a wide range of information about individuals, with the ultimate goal of piloting elections' results, in particular Brexit and presidential elections in the United States. In fact, knowing information such as personality, age, gender, occupation, political and religious affiliation allows to target voters with extremely specific messages that are very likely to be effective. Whether the charge is true or not, it is not clear yet, nevertheless this is a perfect example of malicious use of social network data. Also, MyPersonality project is involved in the scandal, as the tool used by Cambridge Analytica to predict users' personal information was inspired by the work of Kosinski et al. [1, 28]. The company tried with no success to acquire that software, and so mimicked their methods. The researchers probably chose not to cooperate with the company because they were only interested in the analysis of social network data for scientific research, rather than unethical applications.
Analogously, we use myPersonality dataset collected by Kosinski and Stillwell to investigate about whether the personality of an individual can be inferred just by relying on what he writes on Twitter, and we are only interested in the scientific aspect of the research.

Bibliography

[1] Kosinski, M., Stillwell, D., Graepel, T. Private traits and attributes are predictable from digital records of human behavior. *PNAS*, *2013*, *110 (15)*, 5802-5805.

[2] Liu, L., Preotiuc-Pietro, D., Riahi Samani, Z., Ebrahimi Moghaddam, M., H. Ungar, L. Analyzing Personality through Social Media Profile Picture Choice. *International AAAI Conference on Web and Social Media (ICWSM)*, *2016*.

[3] Stutzman, F., Gross, R., Acquisti, A. Silent Listeners: The Evolution of Privacy and Disclosure on Facebook. *Journal of Privacy and Confidentiality*, *2012*, 4(2):7-41.

[4] Barrick, M., Mount, M. The Big Five personality dimensions and job performance: A meta-analysis. *Personnel psychology*, *1991*, 44(1):1-26.

[5] Saulsman, L., Page, A. The five-factor model and personality disorder empirical literature: A meta-analytic review* 1. *Clinical Psychology Review*, *2004*, 23(8):1055–1085.

[6] Huang, Y., Wei, L., Chen, Y. Detection of the Prodromal Phase of Bipolar Disorder from Psychological and Phonological Aspects in Social Media. *2017*.

[7] Shaver, P., Brennan, K. Attachment styles and the "Big Five" personality traits: Their connections with each other and with romantic relationship outcomes. *Personality and Social Psychology Bulletin*, *1992*, 18(5):536.

[8] Rentfrow, P. and Gosling, S. The do re mi's of everyday life: The structure and personality correlates of music preferences. *Journal of Personality and Social Psychology*, *2003*, 84(6):1236–1256.

[9] Dollinger, S. Research Note: Personality and Music Preference: Extraversion and Excitement Seeking or Openness to Experience? *Psychology of Music*, *1993*, 21(1):73.

[10] Hansen, C., Hansen, R. Constructing personality and social reality through music: Individual differences among fans of punk and heavy metal music. *Journal of broadcasting & electronic media*, *1991*, 35(3):335–350.

[11] Rawlings, D., Ciancarelli, V. Music preference and the five-factor model of the NEO Personality Inventory. *Psychology of Music*, *1997*, 25(2):120.

[12] Jost, J., West, T., Gosling, S. Personality and ideology as determinants of candidate preferences and Obama conversion in the 2008 US presidential election. *Du Bois Review: Social Science Research on Race*, *2009*, 6(01):103–124.

[13] Cantador, I., Fernandez-Tobias, I., Bellogín, A., Kosinski, M., Stillwell, D. Relating Personality Types with User Preferences Multiple Entertainment Domains. *CEUR Workshop Proceedings*, *2009*, 997.

[14] Gottschalk, L. A.; Gleser, G. C. The measurement of psychological states through the content analysis of verbal behavior. *University of California Press*; Oxford, England, 1969.

[15] Graham, D. T.; Stern, J. A.; Winokur G. Experimental investigation of the specificity of attitude hypothesis in psychosomatic disease. *Psychosomatic Medicine*, *1958*, *20*, 446–457.

[16] Mergenthaler, E. Emotion-abstraction patterns in verbatim protocols: A new way of describing psychotherapeutic processes. *Journal of Consulting and Clinical Psychology*, *1996*, 64:1306–1315.

[17] Pennebaker, J., King, L.A. Linguistic Styles: Language Use as an Individual Difference. *Personality and Social Psychology*, *1999*, 77(6):1296-1312.

[18] Argamon, S., Dhawle, S., Koppel, M., Pennebaker, J. Lexical predictors of personality type. *Proceedings of the 2005 Joint Annual Meeting of the Interface and the Classification Society of North America*, *2005*.

[19] Celli, F., Lepri, B., Biel, J., Gatica-Perez, D., Riccardi, G. The workshop on computational personality recognition 2014. *Proceedings of ACMMM*, *2014*, 1245-1246.

[20] Tkalčič, M., de Carolis, B., de Gemmis, M., Odić, A., Košir, A. Preface: EMPIRE 2014. *Proceedings of the 2nd Workshop Emotions and Personality in Personalized Services (EMPIRE)*, *2014*.

[21] Hughes, D., Rowe, M., Batey, M., Lee, A. A tale of two sites: Twitter vs. Facebook and the personality predictors of social media usage. *Computers in Human Behavior*, *2011*, 28:561-569.

[22] Bachrach, Y., Kosinski, M., Graepel, T., Kohli, P., Stillwell, D. Personality and Patterns of Facebook Usage. *Proceedings of the 3rd Annual ACM Web Science Conference, WebSci'12*, *2012*.

[23] Gosling, S.D., Augustine, A.A., Vazire, S., Holtzman, N., Gaddis, S. Manifestations of Personality in Online Social Networks: Self-Reported Facebook-Related Behaviors and Observable Profile Information. *Cyberpsychology, behavior and social networking*, *2011*, 14:483-8.

[24] Quercia, D., Kosinski, M., Stillwell, D., Crowcroft, J. Our Twitter Profiles, Our Selves: Predicting Personality with Twitter. , 180-185. 10.1109/PASSAT/SocialCom.2011.26.

[25] Jusupova, A., Batista, F., Ribeiro, R. Characterizing the Personality of Twitter Users based on their Timeline Information. *Atas da 16 Conferência da Associacao Portuguesa de Sistemas de Informação*, *2016*, 292-299.

[26] Liu, F., Perez, J., Nowson, S. A Language-independent and Compositional Model for Personality Trait Recognition from Short Texts. *2016*.

[27] Van de Ven, N., Bogaert, A., Serlie, A., J. Brandt, M., J.A. Denissen, J. Personality perception based on LinkedIn profiles. *Journal of Managerial Psychology*, *2017*, 32.

[28] YouYou, W., Kosinski, M., Stillwell, D. Computer-based personality judgments are more accurate than those made by humans. *Proceedings of the National Academy of Sciences of the United States of America*, *2014*, 112, 10.1073/pnas.1418680112.

[29] Nowson, S., Oberlander, J. The Identity of Bloggers: Openness and gender in personal weblogs. *AAAI Spring Symposium, Computational Approaches to Analysing Weblogs.*, *2006*, 163-167.

[30] Kalghatgi, M.P., Ramannavar, M., Sidnal, N.S. A neural network approach to personality prediction based on the bigfive model. *International Journal of Innovative Research in Advanced Engineering (IJIRAE)*, *2015* 2(8):56–63.

[31] Su, M., Wu, C., Zheng, Y. Exploiting turn-taking temporal evolution for personality trait perception in dyadic conversations. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *2016*, 24(4):733–744.

[32] Ling, W., Luís, T., Marujo, L., Astudillo, R., Amir, S., Dyer, C., W. Black, A., Trancoso, I. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. *2015*.

[33] Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A. Advances in Pre-Training Distributed Word Representations. *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, *2018*.

[34] Majumder, N., Poria, S., Gelbukh, A., Cambria, E. Deep Learning-Based Document Modeling for Personality Detection from Text. *IEEE Intelligent Systems*, *2017*, 32:74-79.

[35] Mikolov, T., Chen, K., Corrado, G.S., Dean, J. Efficient Estimation of Word Representations in Vector Space. *2013*.

[36] Mairesse, F., Walker, M., Mehl, M., Moore, R. Using Linguistic Cues for the Automatic Recognition of Personality in Conversation and Text. *J. Artificial Intelligence Research*, *2007*, 30. 457-500.

[37] Pasca, M.,Lin, D., Bigham,J., Lifchits, A., Jain, A. Names and Similarities on the Web: Fact Extraction in the Fast Lane. *ACL-44*, *2006*,

[38] Manning, C., Raghavan, P., Schtze, H. Introduction to Information Retrieval. *Cambridge: Cambridge University Press.*, *2008*.

[39] Shutze, H. Distributional part-of-speech tagging. *ACL*, *1995*, 141-148.

[40] Ratinov, L and Roth, D. Design challenges and misconceptions in named entity recognition. *CoNLL*, *2009*, 147-155.

[41] Kuang, S., Davison, B. Learning Word Embeddings with Chi-Square Weights for Healthcare Tweet Classification. *Applied Sciences*, *2017*, 7(8):846.

[42] Yang, X., Macdonald, C., Ounis, I. Using Word Embeddings in Twitter Election Classification. *Information Retrieval Journal*, *2016*.

[43] Pennington, J., Socher, R., Manning, C. Glove: Global Vectors for Word Representation. *EMNLP*, *2014*, 14:1532-1543.

[44] Lebret, R., Legrand, J., Collobert, R. Is deep learning really necessary for word embeddings? *NIPS Deep Learning Workshop*, *2013*.

[45] Dhillon, P.S., Foster, D., Ungar, L. Multi-view learning of word embeddings via cca. *Advances in Neural Information Processing Systems*, *2011*, 24.

[46] Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., Qin, B. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification. *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, *2014*, 1. 1555-1565.

[47] Collobert, R., Weston, J., Bottou, L., Karlen, M. Kavukcuoglu, K., Kuksa, P. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *2011*, 12:2493–2537.

[48] Zou, W.Y., Socher, R., Cer, D., Manning, C.D. Bilingual word embeddings for phrase-based machine translation. *EMNLP*, *2013*.

[49] Golbeck. J., Robles, C., Turner, K. Predicting Personality with Social Media *Conference on Human Factors in Computing Systems - Proceedings*, *2011*, 10:253-262.

[50] Jiang, L., Yu, M., Zhou, M., Liu, X., Zhao, T. Target-dependent twitter sentiment classification. *ACL*, *2011*, 1:151-160.

[51] Hu, X. Tang, J., Gao, H., Liu, H. Unsupervised sentiment analysis with emotional signals. *Proceedings of the International World Wide Web Conference*, , 607-6

[52] Mohammad, S.M., Kiritchenko, S., Zhu, X. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *Proceedings of the International Workshop on Semantic Evaluation*, *2013*.

[53] Kanavos, A., Nodarakis, N., Sioutas, S., Tsakalidis, A., Tsolis, D., Tzimas, G. Large Scale Implementations for Twitter Sentiment Classification. *Algorithms*, *2017*, 10(1):33.

[54] Dai, H., Touray, M., Jonnagaddala, J., Shabbir, S.A. Feature Engineering for Recognizing Adverse Drug Reactions from Twitter Posts. *Information*, *2016*, 7(2).

[55] Chamberlain, B.P., Humby, C., Deisenroth, M.P. Probabilistic Inference of Twitter Users' Age Based on What They Follow. *Machine Learning and Knowledge Discovery in Databases*, *2017*, 191-203.

[56] Zhang, J., Hu, X., Zhang, Y., Liu, H. Your Age Is No Secret: Inferring Microbloggers' Ages via Content and Interaction Analysis. *ICWSM*, *2016*.

[57] Burger, J.D., Henderson, J., Kim, G., Zarrella, G. Discriminating gender on Twitter. *EMNLP*, *2011*, 1301-1309.

[58] Conover, M.D., Gonalves, B., Ratkiewicz, J., Flammini, A., Menczer, F. Predicting the political alignment of Twitter users. *PASSAT*, *2011*, 10:192-199.

[59] Cheng, Z., Caverlee, J., Lee, K. You are where you tweet: a content-based approach to geo-locating Twitter users. *International Conference on Information and Knowledge Management, Proceedings.*, *2010*, 759-768.

[60] Pennacchiotti, M., Popescu, A.M. A machine learning approach to twitter user classification *ICWSM*, *2011*, 11.

[61] R. McCrae, R., P. John, O. An Introduction to the Five-Factor Model and Its Applications. *Journal of personality*, *1992*, 60(2):175-215.

[62] P. John, O., Angleitner, A., Ostendorf, F. The lexical approach to personality: A historical review of trait taxonomic research. *European Journal of Personality*, *1988*, 2(3):171-203.

[63] Caprara, G., Cervone, D. Personality: Determinants, Dynamics, and Potentials. *Cambridge: Cambridge University Press*, *2000*.

[64] John, O. P., Robins, R. W., Pervin, L. A. Handbook of Personality: Theory and Research, Third Edition. *New York: The Guilford Press*, *2000*, 114-158.

[65] Galton, F. Measurement of Character. *Fortnightly Review*, *1884*, 36:179-185.

[66] Mershon, B., Gorsuch, R. Number of Factors in the Personality Sphere: Does Increase in Factors Increase Predictability of Real-Life Criteria?. *Journal of Personality and Social Psychology*, *1988*, 55:675-680.

[67] V. Paunonen, S., Ashton, M. Big Five Factors and Faces and the Prediction of Behavior. *Journal of Personality and Social Psychology*, *2001*, 81:524-539.

[68] Costa, P.T. and McCrae, R.R. Revised NEO Personality Inventory (NEO Pl-R) and NEO Five-Factor Inventory (NEO-FFI) Professional Manual. *Odessa, FL: Psychological Assessment Resources*, *1992*.

[69] Likert, R. A Technique for Measurement of Attitudes. *Archives of Psychology*, *1932*, 140:1-55.

[70] Goldberg, L. The development of markers for the Big-Five factor structure. *Psychological assessment*, *1992*, 4(1):26-42.

[71] W. Allport, G., S. Odbert, H. Trait-Names: A Psycho-lexical Study. *Psychological Monographs*, *1936*, 47(1).

[72] T. Norman, W. Toward an Adequate Taxonomy of Personality Attributes: Replicated Factor Structure in Peer Nomination Personality Ratings. *Journal of abnormal and social psychology*, *1963*, 66(6):574-583.

[73] B. Cattell, R. The description of personality: basic traits resolved into clusters. *The Journal of Abnormal and Social Psychology*, *1943*, 38(4):476-506.

[74] Fiske, D. Consistency of the factorial structures of personality ratings from different sources. *Journal of Abnormal Social Psychology*, *1949*, 44:329-344.

[75] M. Digman, J., K. Takemoto-Chock, N. Factors In The Natural Language Of Personality: Re-Analysis, Comparison, And Interpretation Of Six Major Studies. *Multivariate Behavioral Research*, *1981*, 16(2):149-170.

[76] M. Smith, G. Usefulness of peer rating of personality in educational research. *Educational and Psychological Measurement*, *1967*, 27:967-984.

[77] Tupes, E. C, Christal, R. E. (1961). Recurrent personality factors based on trait ratings (USAF ASD Tech. Rep. No. 61-97) *Lackland Air Force Base, TX: U.S. Air Force*, *1961.*

[78] P. John, O. The "Big Five" Factor Taxonomy: Dimensions of Personality in the Natural Language and in Questionnaires. *L. A. Pervin (Ed.), Handbook of personality: Theory and research*, *1990*, 66-100.

[79] Goldberg, L. R., McReynolds, P. A historical survey of personality scales and inventories. *Advances in psychological assessment, Volume 2. Palo Alto, CA: Science and Behavior Books*, *1971.*

[80] C. G. Jung, Psychological Types, translated by H. G. Baynes, revised by R. F. C. Hull. *Princeton University Press*, *1971.*

[81] Murray, H. Explorations in Personality. *New York: Oxford University Press*, *1938.*

[82] The interpersonal theory of psychiatry *New York: Norton*, *1953.*

[83] Tellegen, A., Waller, N. G. Exploring personality through test construction: Development of the Multidimensional Personality Questionnaire. In S. R. Briggs & J. M. Cheek (Eds.). *Personality measures: Development and evaluation (Vol. 1)*, Greenwich, CT: JAI Press.

[84] Costa. P. T. Jr., McCrae, R. R. Age differences in personality structure: A cluster analytic approach. *Journal of Gerontology*, *1976*, 31(5):564-70.

[85] Digman, J. M. The five major domains of personality variables: Analysis of personality questionnaire data in the light of the five robust factors emerging from studies of rated characteristics *annual meeting of the Society of Multivariate Experimental Psychology*, Los Angeles, *1973.*

[86] Hogan, R. Socioanalytic Theory of Personality. *Nebr Symp Motiv.*, *1983*, 55-89.

[87] Leary, T. Interpersonal diagnosis of personality: a functional theory and methodology for personality evaluation. *New York: Ronald Press*, *1957.*

[88] Amelang, M., Borkenau, P. Uber die faktorielle Struktur und externe Validitat einiger Fragebogen-Skalen zur Erfassung von Dimensionen der Extraversion und emotionalen Labilitat [On the factor structure and external validity of some questionnaire scales measuring dimensions of extraversion and neuroticism]. , *Zeitschrift fur Differentielle und Diagnostische Psychologi*, 3:119-146.

[89] McCrae, R. R., Costa, P. T., Jr. Updating Norman's "adequate taxonomy": Intelligence and personality dimensions in natural language and in questionnaires. *Journal of Personality and Social Psychology*, *1985*, 49(3):710-721.

[90] R. McCrae, R., Costa, P. Validation of the five factor model of personality across instruments and observers. *Journal of personality and social psychology*, *1987*, 52(1):81-90.

[91] Hogan, R. Personal Communication. *1990*.

[92] McCrae. R. R., Costa, P. T., Jr. Busch, C. M. Evaluating comprehensiveness in personality systems: The California Q-Set and the five-factor model. *Journal of Personality*, *1986*, 54:430-446.

[93] Brand, C.R. Personality Dimensions: An Overview of Modern Trait Psychology, in Nicholson, J. and Beloff, H. (Eds). *Psychology Survey 5, British Psychological Society, Leicester*, *1984*.

[94] Judge T.A., Higgins C.A., Thoresen C.J., Barrick M.R. The big five personality traits, general mental ability, and career success across the life span. *Personnel psychology*, *1999*, 52(3):621–652.

[95] Ryan T., Xenos S. Who uses facebook? An investigation into the relationship between the big five, shyness, narcissism, loneliness, and facebook usage. *Computers in Human Behavior*, *2011*, 27(5):1658-1664.

[96] Braun, M., Hauser, J.R., Liberali, G., Urban, G.L. Website Morphing. *Marketing Science*, *2009*, 28:202-223.

[97] Kalchbrenner, N., Grefenstette, E., Blunsom, P. A Convolutional Neural Network for Modelling Sentences. *ACL*, *2014*.

[98] Kim, Y. Convolutional Neural Networks for Sentence Classification. *EMNLP*, *2014*, 1746-1751.

[99] Socher, R., Perelygin, A., Y. Wu, J., Chuang, J., D. Manning, C., Y. Ng, A., Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. *EMNLP*, *2013*.

[100] P. Turian, J., Ratinov, L., Bengio, Y. Word Representations: A Simple and General Method for Semi-Supervised Learning. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, *2010*, 384-394.

[101] Salton, G., Wong, A., Yang, C.S. A vector space model for automatic indexing *Communications of the ACM*, *1975*, 18(11):613-620.

[102] S. Harris, Z. Distributional Structure. *Word*, *1954*, 10(2-3):146-162.

[103] Sahlgren, M. The distributional hypothesis. *Italian Journal of Linguistics*, *2008*, 20(1).

[104] MCdonald, S., Ramscar, M. Testing the distributional hypothesis: The influence of context on judgements of semantic similarity. *In Proceedings of the 23rd Annual Conference of the Cognitive Science Society*, *2001*, 611-616.

[105] Deerwester, S. Improving information retrieval with latent semantic indexing. *Proceedings of the 51st Annual Meeting of the American Society for Information Science*, *1988*, 36-40.

[106] Bengio, Y., Ducharme, R., Vincent, P. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, *2000*, 3(6):932-938.

[107] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. Distributed representations of words and phrases and their compositionality. *Neural and Information Processing System (NIPS)*, *2013*.

[108] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, *2016*, 5:135-146.

[109] Joulin, A., Grave, E., Bojanowski, P., Mikolov, T. Bag of Tricks for Efficient Text Classification.*European Chapter of the Association for Computational Linguistics (EACL)*, *2016*.

[110] Lebret, R., Collobert, R. Word Emdeddings through Hellinger PCA. *2013*.

[111] Levy, O., Goldberg, Y. Neural word embedding as implicit matrix factorization. *Advances in Neural Information Processing Systems (NIPS)*, *2014*, 3:2177-2185.

[112] Li, Y., Xu, L., Tian, F., Jiang, L., Zhong, X., Chen, E. Word Embedding Revisited: A New Representation Learning and Explicit Matrix Factorization Perspective. *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15)*, *2015*, 3650-3656.

[113] Li, Q., Shah, S., Fang, R., Liu, X., Nourbakhsh, A. Data Sets: Word Embeddings Learned from Tweets and General Data. *2017*.

[114] Globerson , A., Chechik, G., Pereira, F., Tishby, N. Euclidean Embedding of Co-occurrence Data. *The Journal of Machine Learning Research*, *2007*, 8:2265-2295.

[115] Landauer, T.K., Dumais, S.T. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge *Psychological Review*, *1997*, 104:211-240.

[116] Hinton, G.E., Roweis, S.T. Stochastic Neighbor Embedding *Advances in Neural Information Processing Systems (NIPS)*, *2002*, 15:833-840.

[117] van der Maaten, L., Hinton, G. Viualizing data using t-SNE. *Journal of Machine Learning Research*, *2008*, 9:2579-2605.

[118] Levy, O., Goldberg, Y. Linguistic Regularities in Sparse and Explicit Word Representations*In Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL)*, *2014*, 171-180.

[119] Do, C., Y. Ng, A. Transfer learning for text classification. *NIPS*, *2005*, 299-306.

[120] Raina, R., Y. Ng, A., Koller, D. Constructing informative priors using transfer learning. *ICML 2006 - Proceedings of the 23rd International Conference on Machine Learning*, *2006*, 713-720.

[121] Goldberg, L. R., Johnson, J. A., Eber, H. W., Hogan, R., and Ashton, M. C., Cloninger, C. R., and Gough, H. G. The international personality item pool and the future of public-domain personality measures. *Journal of Research in personality*, *2006*, 40:84-96, 1.

[122] Kosinski, M., Matz, S., Gosling, S., Popov, V., Stillwell, D. Facebook as a Social Science Research Tool: Opportunities, Challenges, Ethical Considerations and Practical Guidelines. *American Psychologist*, *2015*.

[123] Celli, F., Pianesi, F., Stillwell, D., Kosinski, M. Workshop on Computational Personality Recognition: Shared Task. *2013*, *AAAI Workshop - Technical Report*.

[124] Kenney, J.F., Keeping, E.S. Linear Regression and Correlation. *Mathematics of Statistics*, *1962*, Ch. 15, Pt. 1, pp. 252-285.

[125] Tibshirani, R. Regression shrinkage selection via the LASSO. *Journal of the Royal Statistical Society Series B*, *2011*, 73:273-282.

[126] Drucker, H., Burges, C.J.C.Kaufman, L., Smola, A.J. Vapnik, V.N. Support Vector Regression Machines. *Advances in Neural Information Processing Systems 9, NIPS* , *1996*, 155-161.

[127] John, O. P., Naumann, L. P., Soto, C. J. Paradigm shift to the integrative Big Five trait taxonomy: History, measurement, and conceptual issues. In O. P. John, R. W. Robins, L. A. Pervin, *Handbook of personality: Theory and research*, *2008*, 114-158.

[128] Back, M., Stopfer, J., Vazire, S., Gaddis, S., Schmukle, S., Egloff, B., Gosling, S. Facebook Profiles Reflect Actual Personality, Not Self-Idealization. *Psychological Science*, *2010*, 21(3):372.

[129] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *2011*, 12:2825-2830.