Universidad Autónoma de Chihuahua

Facultad de Ingeniería

Secretaría de Investigación y Posgrado

Towards new representations of documents for author profiling in digital texts

A DISSERTATION SUBMITTED BY

Jesús Roberto López Santillán

IN FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Engineering

CHIHUAHUA, CHIH., MEXICO. MARCH, 2021

Towards new representations of documents for author profiling in digital texts. A dissertation submitted by Jesús Roberto López Santillán in partial fulfillment of the requirements for the degree of Doctor of Engineering, has been verified and accepted by:

[illegible]

M.I. Javier González Cantú
Director de la Facultad de Ingeniería

[illegible]

Dr. Alejandro Villalobos Aragón
Secretario de Investigación y Posgrado

[illegible]

Dr. Luis Carlos González Gurrola
Dissertation Director

[illegible]

Dr. Manuel Montes y Gómez
Dissertation Co Director
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)

March, 2021

Examination committee:

Dr. Luis Carlos González Gurrola
Dr. Manuel Montes y Gómez
Dr. Raymundo Cornejo García
Dr. Graciela María de Jesús Ramírez Alonso
Dr. Alberto Camacho Ríos

Abstract

Digital Text forensics (DTF) is a novel term coined to deal with investigating truthfulness in digital data banks. It mostly handles authorship analysis such as *plagiarism detection*, *author identification*, *author obfuscation* and *author profiling*. Specifically, Author Profiling (AP) deals with the problem of identifying personal traits of authors within their texts. In order to make a computer understand terms and perform this chore, words must be represented as numerical entities, such as vectors in a n-dimensional space. A way to do this is through Word Embeddings (WE), these can be viewed as high density vectors that code the meaning of words, in such a way that similar concepts tend to cluster. The use of WEs deliver State of the Art (SoA) results in various Natural Language Processing (NLP) tasks such as text classification. Nonetheless, for problems like AP or Sentiment Analysis (SA), word vectors might not be sufficient to attain good results. The reason behind this lacking in better results can be attributed partially to WEs being non-contextual. To tackle the latter problems enhanced embeddings must be produced, to improve the context prowess of WEs or even to represent whole documents instead of single words. Theoretically, these vectors might have an even higher density and are known as *Document Embeddings* (DE). Such representations attempt to capture *"intention"* within documents, so they might be useful in the aforementioned AP and SA tasks. The idea behind DEs is that a document may be viewed as the aggregation of its words, this means that a new vector can be composed for example by averaging the WEs of its terms, to produce a single vector that describes the whole document. The problem with a simple averaging of terms relies within the importance of words. To compose a DE not all terms must be aggregated with the same weight. Different strategies can be useful to make more emphasis on some words; take for instance the *term frequency-inverse document frequency* (tf-idf), which assigns values to words depending on the frequency that they appear on texts, offset by the rarity of the documents they appear on. Moreover, custom weighting metrics can be devised depending on the task. In the first part of the present thesis a new statistic is introduced to establish importance of words: the *relevance topic value (rtv).* Likewise, *Genetic Programming* (GP) is used as an evolutionary strategy along with these statistic metrics to produce term weighting schemes. The aforementioned strategy has been successfully tested in 17 AP tasks over nine datasets, attaining top-quartile results compared to official results of six shared tasks held at the scientific event called *Uncovering Plagiarism, Authorship and Social Software Misuse* (PAN) over the years 2013-2018. In the second part of this dissertation, a novel and specialized Deep Neural Network (DNN) architecture is proposed to tackle the AP problem. This new DL method merges concepts of *Wide & Deep* (WD) networks, *self-attention* and the *Transformer* architecture, which is called *The Profiler* aka the *Wide & Deep Transformer* (WD-T). This approach addresses AP tasks by noticing personal and stylistic latent information useful to discern characteristics of authors in social media posts. Then, encodes this information as *wide* features, which are inputted along with *deep* contextualized WEs into the WD-T. Although this method was expressly tested in AP tasks, by the achieved results, it can be inferred that different NLP problems might be successfully addressed using this architecture with different *wide* features. The *Profiler* (WD-T) was tested in five AP tasks of the 2017, 2018 and 2019 PAN datasets. Historically the best performers at such shared tasks implement traditional Machine Learning (ML) approaches. Even so, the WD-T was able to achieve results in the top-quartile positions overall, while outperforming all DL participants in those competitions. In short, this thesis addresses the AP problem using novel methods, based on traditional and deep ML approaches. In both cases, the proposed methodologies attained competitive results and demonstrates the rationale behind the idea, thus contributing to the SoA in the AP/NLP field.

For Olanda, Natalia & Ana Lucia

"You want to know how I did it? This is how I did it, Anton:
I never saved anything for the swim back."
Vincent.
Gattaca (1997)

Agradecimientos

La culminación de mis estudios de doctorado no hubiese sido posible sin el inconmensurable apoyo de muchas personas e instituciones. Reconociendo la potencial ingratitud de omitir a alguno de estos me atrevo a nombrarlos.

A la Universidad Autónoma de Chihuahua (UACH), mi siempre alma mater a la cual estoy profunda e infinitamente agradecido y orgulloso de pertenecer, por el apoyo brindado desde hace ya más de 20 años.

Al Dr. Luis Carlos González Gurrola mi director de tesis. Por su guía y apoyo incondicional. Por creer en mi cuando incluso yo mismo dudaba, pero sobre todo por su sincera amistad.

Al Dr. Manuel Montes y Gómez mi co-director de tesis. Gracias por todas las ideas y esas largas y gratas tertulias. Por invitarme a una estancia en el Instituto de Astrofísica, Electrónica y Óptica (INAOE), la cual me dio grandes experiencias no solo en el ámbito científico. Pero sobre todo por la amistad.

A mi comité doctoral, la Dra. Graciela María de Jesús Ramírez Alonso, el Dr. Raymundo Cornejo García y el Dr. Alberto Camacho Ríos. Por la guía que nunca escatimaron, las enseñanzas y consejos que perdurarán conmigo por siempre.

Al Dr. Alejandro Villalobos Aragón, por su amistad y palabras de aliento en momentos complicados.

A la Maestra Luly Flores, por su infinita paciencia y amabilidad.

Al Dr. Leonardo Trujillo Reyes del Instituto Tecnológico de Tijuana, por su invaluable apoyo en las etapas tempranas del doctorado.

A la Coordinación General de Tecnologías de Información de la UACH, cuyos recursos computacionales fueron fundamentales para la elaboración de los experimentos de esta tesis.

Al Instituto de Astrofísica, Electrónica y Óptica (INAOE) y al Dr. Luis Villaseñor Pineda, responsable de la *Plataforma de Aprendizaje Profundo para Tecnologías del Lenguaje* del Laboratorio de Supercómputo del INAOE, por darnos acceso a recursos computacionales extraordinarios, fundamentales en el desarrollo de este trabajo.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT), que a través de los proyectos de Fronteras de la Ciencia: *"Fuerza de atracción textual: Hacia un nuevo paradigma de clasificación de documentos"* FC-2410 2017-2019 y *"Aprendizaje evolutivo a gran escala"* FC-2015-2:944, hicieron posible la elaboración de gran parte de esta tesis.

A mi padre, quien me heredó la curiosidad y la pasión por la investigación.

A mi madre, de quien obtuve mi amor por la docencia.

A mi hermano, quien a pesar de ser menor que yo me da lecciones invaluables.

A mi hermana, por ser una luz en oscuros momentos, y por su ejemplo de valentía y tenacidad.

A mi esposa, por ser mi cimiento, mi inspiración y mi cómplice, pero sobre todo por soportar y apoyar todos mis caprichos.

A mis hijas Natalia y Ana Lucia, por ser mis musas y mi razón de vivir.

A Dios, por todo....

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Author Profiling in the context of Digital Text Forensics

Digital Text Forensics (DTF) is a novel term coined to deal with investigating truthfulness in digital data banks. It mostly handles authorship analysis such as *plagiarism detection*, *author identification*, *author obfuscation* and *author profiling* [1]. Specifically, Author Profiling (AP) is a computational task which aims to predict personal characteristics of authors (e.g., gender, age or even personality traits) based on texts that authors themselves have produced. The variety of these texts could span examples of formal writing, such as essays or articles, to informal interchange like comments in social media or even product opinions in reviews [2]. AP has become popular as a Natural Language Processing (NLP) task, given that it has proved its value in sensitive applications like digital security, spotting Internet predatory activities or detecting fraud and cyber-terrorism [3, 4]; but also it has been applied to improve customer service, diagnosis of neurological disorders (e.g., autism, depression) or detection of plagiarism [5, 4, 6, 7].

1.2 Representation of texts for author profiling

To identify personal features of authors it must be examined how they use words. A way to computationally manipulate these words is by representing them as numeric vectors. In this sense, Word Embeddings (WEs) can be viewed as high density vectors that encode the meaning of words, in such a way that similar concepts tend to cluster within a n-dimensional space, hence the distance between WEs is a measure of similarity [8, 9]. Consider that WEs have delivered State of the Art (SoA) results in tasks such as *text classification*, *language translation* or *speech recognition* [8]. This idea could be extrapolated to represent bigger chunks of text (commonly known as *Document Embeddings* (DE)), and can be used to extract useful information, such as "intention" from a whole document, an approach that has been exploited for AP and Sentiment Analysis (SA) tasks [10]. DEs can be produced by several techniques, being the *centroids method* one of the most popular and successful [11]. The idea behind the *centroids method* is that a sentence, paragraph or even a whole document can be viewed as the aggregation of its words, therefore a DE could be generated by averaging the WEs of terms contained in the document.

1.3 The shortcomings of current text representation techniques

Since it is based on the average concept, the *Centroid* method could present some shortcomings to capture subtle differences in authors writing styles, thus making it not entirely appropriate as it is for AP endeavors. To overcome this issue, in the first part of this thesis a novel strategy is presented to generate DEs. This proposal relies on the hypothesis that it is possible to find novel and optimized weights for each word within a document, thus producing an improved aggregate strategy instead of just averaging terms. To test this hypothesis Genetic Programming (GP) was employed, which is a very sound approach to learn intrinsic structure within data via mathematical equations [12]. According to the most recent literature review, GP has not been employed for the purpose of evolving weighting schemes to aggregate WEs into DEs in the context of AP, so a new application of GP is envisioned. The first proposed pipeline is as follows: GP employs statistical features for each word within a document (e.g., *term frequency* (tf), *term frequency-inverse document frequency* (tf-idf)) to evolve equations to calculate the weights (importance) of terms. Then, using word vector algorithms (e.g., word2vec, fastText, BERT), WEs are produced for terms in the datasets. Next, WEs from users posts are aggregated into DEs using a weighted average (importance established by GP). Finally, using a Machine Learning (ML) approach the DEs are used as features to predict the *gender*, *age*, *language variety* and *personality* of authors. In addition, a novel numeric statistic feature (*rtv*) is introduced, which is based on a frequency analysis over the use of words and themes by persons. Moreover, *rtv* turned out to be the most likely feature to appear alone in a single equation, then suggesting its usefulness as a WE weighting-scheme factor.

1.4 The role of task-specific feature engineering

For the second part of this dissertation another approach is introduced, based on a Deep Learning (DL) architecture known as the *Transformer*, which simplifies the structure of Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) to introduce an only-attention mechanism. In addition, inspired by the ideas of Cheng et al. in [13], a joint effort of *wide* and *deep* learning was explored. The rationale behind the idea is that combining the generalization prowess of deep networks with the task-specific feature engineering (*wide* features), into a *wide & deep* architecture, could render a better model that produces more accurate predictions. That is the case of tasks such a AP. Hence, a novel DL architecture is proposed specifically for the AP problem, called *The Profiler* aka the *Wide & Deep Transformer* (WD-T). This architecture, makes use of a structure similar to the *encoder* section of a *transformer*, but dispenses of the multi-head attention structure in favor of a single self-attention block, for contextualizing WEs produced by the skipgram method (word2vec) [8], from datasets expressly designed for AP endeavours. In addition, a novel feature engineering methodology was devised for capturing fine grained characteristics in texts, closely related to the profile of an author, thus enhancing the predictive dexterity of the WD-T for AP tasks, by encoding these features in a *wide* block that jointly with the *deep* encoder attains very competitive results.

1.5 Different approaches to the same task

The first proposal was exhaustively evaluated over a total of nine datasets (in 17 tasks) that were originally devised for the scientific event called *Uncovering Plagiarism,* *Authorship and Social Software Misuse* (PAN[1]) [15], within the period 2013-2018. Each year, the *Conference and Labs of the Evaluation Forum* (CLEF) organizes the PAN conference including a shared task in AP, where several teams compete to predict author's features such as gender, age or personality traits (e.g., openness, extroversion, etc.) originated in a variety of multilingual social media sources. Within this context, the first approach could be contrasted against all the teams that submitted an entry for each year's contest. For completeness, two averaging baselines: a) *a weighted average using only tf-idf values*, and b) *using a simple mean of the WEs (centroids)* were also included. The results of each comparison show that the proposed approach offers very competitive performance to solve AP related tasks, ranking in the top-quartile in every year's competition. These results also suggest the flexibility and robustness of this methodology, since through all the yearly competitions different AP tasks and datasets have been used.

The second strategy was also tested using the same datasets, but this time just a subset of those were employed, focusing only in English and Spanish datasets of the 2017-2019 shared tasks. The reason for selecting only these datasets was a greater participation of DL strategies than in previous events, so a more consistent comparison could be made. The AP tasks consisted in forecasting *gender*, *language variety* (e.g., British English, Mexican Spanish), and identification of *bots* from *humans* for the 2019 competition. Historically, in these shared tasks the top-quartile achievers tend to employ more "traditional" approaches of ML, whilst the DL methods lean to the lack luster side. In this context, it is worthy of mention that the results achieved by the WD-T scored in the top-quartile positions overall and outperformed all previously used DL based techniques.

1.6 Problem statement

In this day and age of mostly real life online interactions, DTF are fundamental for a satisfactory operational environment. Within the DTF, the AP problem is of great interest from the security and marketing point of view [1]. A fair amount of works have attempted to solve the AP problem, for example several share tasks events are

[1]This acronym comes from the title of the first workshop held at SIGIR-2007: Plagiarism analysis, Authorship identificatioa quen, and Near-duplicate detection [14]

held each year to test accuracy of new algorithms [15]; these works report competitive results predicting the gender of individuals, but they struggle in the forecasting of personality traits like *Openness (to experience)*, *Conscientiousness*, *Extraversion*, *Agreeableness* and *Neuroticism* as stated by the *Five Factor Model* (FFM) of personality [16]. ML is a computer science field that has re-gained strength in the last years with the advent of DL, a subfield of ML that has obtained strong achievements in image & speech recognition, computer vision and as of lately Natural Language Processing (NLP). NLP deals with the problem of how computers can understand Human natural language [17]. AP may be viewed as a sub-task of NLP, and it should be approached as such. The AP task is a difficult one, mainly because involves several disciplines (e.g., computer science, linguistics, psychology). Nonetheless, with ML tools available nowadays, it is a problem that can be addressed with high hopes of finding a sound solution, from the application point of view.

1.6.1 Variable nature of author profiling tasks

As already mentioned, the PAN scientific event held each year organizes several NLP tasks, being AP one of the main attractions [18]. In this regard, several obstacles need to be addressed. For example, the profiling of features like gender, age and personality traits must be approached differently, meaning that gender classification might be easier than predicting the "openess" trait. The success of an AP task could be measured in terms of how thorough an specimen can be profiled [16, 19].

1.6.2 The frailty of deep learning in author profiling

Another hurdle to overcome in AP is related with the "underperforming" of novel DL methodologies, which might be attributed in part to the size of available datasets. Mostly "traditional" approaches have attained the top-quartile results since the 2013 PAN competition. Nonetheless, in recent years DL approaches have been gaining terrain in the field of NLP. Take for instance that a RNN can be used to generate text in the style of Shakespeare[2] with incredible results. The documented effectiveness of RNNs on NLP tasks motivate a wider exploration of their reaches on problems like

[2]http://karpathy.github.io/2015/05/21/rnn-effectiveness/

AP. Literature states that RNNs, CNNs and as of lately Transformers are the most employed DL architectures for NLP tasks [20].

1.6.3 Limitations of word vectors

Although WEs deliver SoA results in NLP tasks such as text classification, more difficult assignments like SA (which tries to identify positive from negative opinions) or AP, are not benefited in the same way. WEs capture the "meaning" of a term, nonetheless they still are non-contextual [21]. This means for instance that a unique vector for the word "bank" is produced, whether the intended meaning is either a *financial institution* or a *river bank*. Hence, the averaging of WEs to produce DEs (*centroids* method) for identifying the gender, the age or even personality traits of persons behind digital manuscripts is limited.

1.6.4 Novel effective proposals

In the present work, departing from traditional NLP strategies, different techniques were attempted to solve the AP problem. Two approaches are proposed, which make use of the basic concepts of *WEs*, *GP*, *DL*, *attention mechanisms* and *wide & deep* networks; for contributing to the SoA in *feature engineering* and *learning methods* in the context of AP and NLP. These methodologies were able to capture *stylistic features* (SF) of authors from natural language datasets, in order to accurately predict some of their characteristics (e.g., gender, age). The main premise of the current thesis relies on the theory that the written style of authors play a decisive role in their profiling. SFs are determined by a number of techniques for arranging words in a document, so that the intended message from the author can be properly delivered. The style of an author is often determined by the use of such techniques [22]. The reader might wonder the reason to approach the same problem from two different perspectives. The argument could be constructed from the point of view of application and resources. "Ttraditional" ML does not need a huge amount of data to perform reasonably well. At the same time, it can offer a sound solution when computational resources are scarce. On the other hand, DL struggles when there is a shortage of information, whilst needing a considerable amount of computational resources to perform adequately. Furthermore, many times the type of problem at hand is the only criterion for selecting one strategy or the other. Hence, in this work two different but reliable and proven methodologies are presented. The first one represents a technique that could be considered "traditional" ML, whereas the second belongs to the family of DL architectures.

1.7 Research questions

In order to guide the present work, research questions are stated as follows:

- Could learned weighting-schemes of terms be applied to build DEs from WEs, and achieve competitive results in a variety of datasets for AP tasks?
- Is there a way to combine the attention mechanism of a *Deep Learning* architecture, and the fine grained perception of a wide network, over textual datasets and their latent information (e.g., statistic and/or stylographic features of texts), so it can perform competitively in AP tasks?

In the interest of fulfilling the research questions, the objectives of this thesis are:

1.8 General Objectives

- To devise a novel approach to produce DEs in order to predict more accurately characteristics of authors such as gender, age and personality traits. By building task-specific weighting-schemes that are learned from each particular dataset.
- To conceive a novel *Deep Learning* architecture that makes use of attention mechanisms and wide & deep configurations, for DTF applications (specifically tested in AP tasks).

1.8.1 Specific objectives

- To implement an evolutionary approach (GP) to evolve mathematical equations, that could act as practical task-specific weighting-schemes of terms, to produce DEs for the AP problem.

- To design a new term-importance statistical value for the AP task, that could be used along with well known NLP statistics (e.g., *tf-idf*), to serve as a terminal variable in GP for evolving formulas (weighting-schemes), with the aim to capture how persons employ words and themes.
- To devise a novel *Deep Learning* architecture expressly for AP tasks, based on attention mechanisms and wide & deep models, that could properly achieve the profiling of authors in social media posts.

1.9 Significance and contribution

The contributions of the present thesis are:

- A set of novel approaches to compete in DTF tasks, specifically in AP.
- A novel statistic measure to ponder the importance of terms within a document known as *relevance topic value (rtv)*
- A custom evolutionary approach based on *Genetic Programming* to produce a weighting scheme of terms in a document.
- A novel DL architecture based on attention mechanisms, the transformer architecture (encoder) and wide & deep networks, that could be used in DTF applications, expressly in AP tasks.

1.10 Outline of the dissertation

In **Chapter 2**, several related research works are described, which have addressed similar problems to those posed in the current thesis. Next, in **Chapter 3** various theories behind the several techniques used to implement the proposed methodologies are detailed. Then, in **Chapter 4**, the heuristic approach employed to comprehensively address the AP problem through out 2013-2018 datasets is deconstructed. Moreover, in **Chapter 5** the novel DL technique known as the *Profiler* (WD-T) based on *attention mechanisms*, the *transformer* architecture and *wide & deep* networks, will be explained in detail. Finally in **Chapter 6**, several general conclusions about the reach of this dissertation are elaborated, as well as the future work envisioned.

Chapter 2

Literature Review

Through this chapter the lector will learn about the SoA in some NLP sub fields closely related with the AP problem. Many NLP tasks are similarly approached, thus it becomes relevant knowing how researchers have applied their methodologies to address these problems. In doing so, it will become clear that both methodologies proposed in this work are novel, effective and pertinent.

2.1 Social Media or Thought Contagion?

The way to approach society's needs has evolved through out the *social media* phenomenon, now they must be treated as a product of *collective thinking*. This means that it might be difficult to discern between individual belief and massive opinions. In this regard, social media analytics has become fundamental for modern-life activities such as *product placement*, *massive SA*, *political marketing*, *stock market prediction* and even real-time *information retrieval* [23, 24]. In order to make educated decisions, trendings must be evaluated and assessments be carried out. At the same time, several challenges need to be addressed such as the wide spread of *fake news*, *propaganda*, *disinformation* and information pollution in general [25, 26].

Various recent works have attempted to deal with this current set of problems. Take as an example the research of Chunlin et al., where the authors proposed in [27] an algorithm that helps *community detection* within social networks. Moreover, Automatic Language Identification (ALI) is the cornerstone of successful social media mining. In this matter, Sarma et al. devised a method to perform effective ALI from social media outlets [28]. Furthermore, establishing user profiles from social networks is useful in recommender systems. In this sense, Sanchez et al. introduced in [29] a method that delivers competitive results against cutting edge recommendation techniques. In addition, nowadays social networks such as Twitter act as live breaking news channels. In this subject, Khatuaa et al proposed in [30] a method for training Word Embeddings (WE) over domain-specific datasets, for predicting potential outbreaks of contagious diseases like Ebola or Zika. Although these methods were only tested with traditional ML algorithms, the attained results suggests that if more multilingual medical data is available for training, the potential applications could be of high social impact.

These are only a few examples that show the impact of *social media* and *collective thinking* in modern real life interactions. However they make it clear that a proper solution in the field of AP would greatly aid in said interplays.

2.2 Citizen Erased: The Importance of Author Profiling

Among all NLP related tasks, AP has been profoundly studied in the last decade due to its growing importance. Security and marketing over the Internet are crucial areas, which can be enforced by means of an AP strategy [31]. From the standpoint of safety and protection, predicting the age and gender of authors of social media posts, could help prevention of sexual predatory threats, inhibit potential terrorist attacks, avert financial fraud, among others [4]. In addition, the forecasting of combined characteristics such as age, gender and personality traits, may aid early detection of serious ailments the likes of depression and anxiety. Furthermore, economic activities can be enhanced by using AP techniques, to properly acknowledge and opportunely address customer demands, by anticipating the gender and age of persons behind social media accounts. In this day and age of massive online interaction, the mechanisms to establish one's identity have become fundamental. From online banking proper identification, to online shopping and government procedures, it is crucial to have adequate means to establish correct digital personal profiles. As of today the AP problem has been tackled with promising results, nonetheless, there is still room for improvement [32, 33].

2.3 Author Profiling and the Map of the Problematique

Authorship Attribution (AA) is a task addressed since the Renaissance [34]. Automatic Authorship Identification (AAI) is not a new field either, since it has been studied for more than a hundred years [35]. AAI has been tackled using different techniques like stylometry (i.e., the study of linguistic patterns of authors). The writing style of people may be viewed as a personal signature, which is difficult to forge. This premise allows to predict the profile or even the author itself of written works. Several *identification* problems have been solved by the use of stylometry. For instance, the Federalist papers is a collection of articles devised in the late 1700's to promote the Constitution of the United States of America [36]. The authors of such texts wished to remain anonymous, but in 1944 Douglas Adair attributed the authorship of this articles to Alexander Hamilton, James Madison and John Jay using stylometric techniques. This was confirmed later by a computer analysis in 1964 [37]. In this regard, AAI might be viewed as an immediate antecedent of AP, which ultimately is considered a sub problem. As can be seen, the need for appropriate ways to profile individuals comes from long ago. However, at present its importance has gained more relevance.

2.4 A Globalist Approach to Author Profiling

The identification and profiling of contemporary authors is not only limited to physically published works such as books. Currently, the majority of texts are found online, and profiling and recognition of authors have different applications. For example multilingual social media now represents a huge part of digital entries that might need to be profiled [3]. The current forensic analysis techniques for identifying potential threats, deal with extensive amounts of data, but only in short bursts from each author (e.g., social media posts, blog entries). The size of the texts available on a single author for training is known to affect the outcome of automated classifiers (ML). Even though larger texts are preferred, short samples like the ones found in social media platforms such as Twitter, can be useful in the AP task. Then, several studies show that the accuracy to predict gender and age in short texts declined little when compared to larger samples available, as long as the amount of such specimens is plentiful [16]. Take for instance the work of De Boom et. al in [38], in their study they approached this problem devising a vectorial representation based on WEs which outperformed SoA baselines in semantic similarity tasks. Their method serves well as an out-of-the-box strategy for other tasks, nonetheless it remains unknown if this approach could attain the same results in multilingual of variable length datasets, as the authors only tested their technique on English fixed length data. Moreover, social media can provide additional information, that might help the profiling or identification of authors over the Internet. For instance, Rangel et al. demonstrated in [5] that modeling detected emotions through graphs in posts from social media data banks, helps the prediction of age and gender more accurately. In this study the authors showed evidence that their method could capture more complex language structures. However, their approach did not perform well on short texts. Even though it was tested only in Spanish datasets, it could be inferred a similar performance in other languages too.

Moreover, SA tasks pose a similar difficulty as AP problems, in both cases the representation of the author's "intention" is fundamental, to predict the opinions and characteristics of authors respectively. For example, the works of Saif et al. and Fersini et al., aimed to boost the representation of words through polarity [39, 40]. Although Saif et al. outperformed existing lexicon labeling methods, their approach was not contrasted against more novel ML algorithms, mainly due to the lack of datasets for evaluating entity level sentiments. On the other hand, Fersini et al. proved that their method performs well in cross-domain environments, yet it is language dependant and, as with many ML techniques, does not perform well with a small number of training instances. In AP, increasing the discerning capabilities of features, might be accomplished by accentuating the *personal data* in posts. For example, the study of Ortega-Mendoza et al. demonstrated an increased performance of up to 7% in AP tasks, by emphasizing the *personal information* of authors [41]. Their method demonstrated robustness when used with different ML classifiers. Although their proposed technique only explores personal information as enhancer, it pose the question of whether another type of latent data, could also improve features for AP related tasks. In this regard, both of the methodologies introduced in this thesis, offer a solution to some of the shortcomings of the techniques just described in the past lines. By presenting effective multi-lingual solutions, that also incorporate several types of latent information that increase their overall performance.

2.5 Aiming for Supremacy: Author Profiling as Shared Tasks

Shared tasks are a great way to encourage researches to test their current works. In addition, they promote a continuing improving community that can share knowledge and enrich the area in question. For the AP problem, the PAN scientific event coordinates an annually competition since 2013. Every year new datasets and challenges are introduced, so that in order to solve the tasks, the eventual winner teams need to try different novel approaches. Take for instance the participation of López-Monroy et al. in 2013, they crafted their features on second order representations [42]. In 2014 again López-Monroy et al., now used as features Second Order Attributes (SOA) (a dense and low dimensional representation of documents) [43]. For the 2015 edition, Álvarez-Carmona et al. made a combined use of SOA and Latent Semantic Analysis (LSA) techniques [44]. Furthermore, in 2016 Busger et al. also employed SOA to represent features [45]. Likewise, Basile et al. in 2017 used *tf-idf* weighted character and word n-grams as features [46]. In 2018 Daneshvar et al. performed feature construction (character and word n-grams) and dimensionality reduction using LSA [47]. Finally, in 2019, the task consisted in indentifying *bots* from *humans* and also the *gender*. The winner in this year's shared task employed a character and word n-grams approach for feature engineering, and the Support Vector Machine (SVM) algorithm as a classifier [48]. As can be seen, and also concluded by the organizers in [49], the traditional ML approaches generally attained the top-quartile results. Nonetheless, in recent years (mostly since 2017), DL techniques have been tried out also, but with less than ideal results. Take for instance the 2017 competition, the work of Miura et al. in [50] addressed AP by using a sophisticated DL architecture containing WEs, Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) layers with attention mechanisms. In 2018 even more DL strategies competed. For example, Hosseinia et al. in [51] approached the AP task by using two parallel Long Short-Term Memory (LSTM) RNNs, also with attention mechanisms. Finally, in 2019 more deep networks took part in the shared task. That is the case of Polignano et al. in [52], where they used a CNN. There are many more instances of DL approaches in the most recent competitions, all of which will be examined in *Chapter* 5. Several hypothesis may be established regarding the reason for the underperforming of DL strategies, being the size of the datasets one of various causes, since DL models are known to be successful mainly when huge amounts of training data are available. Furthermore, the variety of approaches (either "traditional" or "deep") shows that there is no gold standard to address AP tasks, so there is plenty of room for improvement. By the end of this thesis, the reader will have learned how the proposed methodologies compare with the competitors just explained, and also will find interesting the high performance achieved by them; attaining in many cases SoA results.

2.6 New Born Trends in Text Representations

One of the main goals of NLP is to construct an effective computational representation of text [53]. Since WEs broke out circa 2013, a huge leap forward was made to depict words in computational terms. Even though WEs have helped the improvement of some NLP tasks, other ones such as AP or SA have yet to benefit in the same manner, thus it is of great importance the continuing search for better text representations techniques. Several works have focused on devising an effective method to represent sentences, starting off from WEs. Take for instance the work of Rossiello et al. in [54]. In their study, the authors aimed to perform text summarization, by building sentence embeddings that represent the abstract of the original text using a centroid-based approach, so the closest sentences to the *centroid* are used to construct the text summarization. Although simple this technique proved to be competitive, yet it shows that centroid-based methodologies in deed capture latent information.

Whilst the aggregation of WEs is the preferred method to construct document Embeddings (DE) for NLP problems, the nature of the task determine the assembly technique to apply. For example Arora et al. used a composition procedure by calculating a *weighted-average* of WEs, using as weight values a measurement similar to *tf-idf* [55]. Granted their technique attains competitive results in *textual similarity* and SA tasks, it is limited when detecting sentiments, mainly due to its generic weighting scheme. In addition, the "geometry" of the DEs has also been studied. For example Michel et al. proposed in their work a topological representation of documents, by "drawing" their "shape" using the euclidean distance between WEs in their n-dimensional space [56]. Although they aimed to boost the performance of DEs in more complex NLP tasks such as SA, the results attained by their study were not entirely satisfactory. Nonetheless, the sophisticated mathematical technique for establishing a geometric shape of DEs, begets further research. Furthermore, the work of Chen in [57] addressed SA and document classification problems also through composing DEs by aggregating WEs. Their proposed approach randomly deleted portions of text for computational tractability, whilst simultaneously enhanced the discerning prowess of *terms* using context, although it is yet to be tested on multilingual data. Moreover, Wu et al. proposed a new type of *sentence embeddings* departing from pre-trained WEs. They employed a *distance* metric known as the *word mover's distance*, to provide their embeddings with contextual vectorial meaning. They attained competitive results in several classification and similarity tasks [58]. These studies demonstrate that task-specific policies for constructing DEs are very useful.

Producing a *weighting-scheme* for terms via an evolutionary method has been proved to be effective. For instance, Escalante et al. proposed in their work a Genetic Programming (GP) approach to produce useful weighting schemes of words, to represent documents in a vector space [59]. Even though the authors did not employ WEs as features, their research attained competitive results against SoA results in text classification tasks, and generalizes well in other NLP problems.

It will be clear in the next chapters how the first proposed methodology of this thesis performed very well against SoA results. By evolving weighting schemes for WEs, that pick up latent information valuable for the profiling of authors.

Moreover, DL and WEs have become a very successful jointly effort in the NLP field [4, 60]. For instance, when aggregating WEs into DEs one major goal is to pay attention to the context of words, to determine their weights. McCann et al. implemented in [61] an attention sequence-to-sequence model to contextualize WEs. Their approach outperformed pre-trained word embeddings (GloVe), as layer starters of deep neural models in several NLP tasks. This study shows promising results as an enhancer of pre-trained or custom WEs, even if they are used as a feature extraction technique, yet it remains to be tested on multilingual datasets, though performance is expected to be in the same level. In addition, the work of Mahdi et. al in [2] demonstrated how to improve pre-trained WEs such as *Word2Vec* and *GloVe* in SA tasks. Nonetheless it is not clear if customized WEs will also benefit in the same way. *Recurrent Neural Networks* (RNN) are the top DL choice for NLP tasks. RNNs are known for their efficacy in time-series type of problems [4]. NLP is often seen as a term sequence problem. In this regard, Palangi et al. used a LSTM network to produce semantic embeddings for information retrieval purposes [62]. Although this approach outperformed the *paragraph vector* algorithm (doc2vec), this study could benefit from a more comprehensive comparison against other methods. Even though RNNs deliver very strong results in domain-specific NLP tasks, for a general-purpose learning approach, Wieting et al. illustrated in [63] that term-averaging still outperforms DL techniques. In addition, RNNs can not implement parallelized learning, due to their sequential nature, thus being inadequate in cases where the volume of information is huge.

Currently, WEs have become the building blocks for representing text features in NLP tasks. Vectorial meaning is one main advantage of these structures, nonetheless being non-contextual entities, somehow limits their performance in several cases. This is why Language Models (LM) have recently emerged, to produce contextual embeddings, trained over larger corpus, using SoA Deep Neural Networks (DNN). Take for instance LMs such as *Embeddings from Language Models* (ELMo) [21] and *Bidirectional Encoder Representations from Transformers* (BERT) [64]. Among several objectives, such as *translation*, *question answering*, *text generation* or even *language understanding*, these new LMs can be employed in the same manner of WEs, to produce contextual word vectors, which have already performed better in several NLP tasks. The second novel proposed model in this thesis work, will demonstrate a *transformer* based architecture that contextualize WEs, whilst encoding latent features which provide a very effective and competitive performance. At the same time, it addresses the shortcomings of RNNs regarding parallel learning, hence presenting a more efficient solution.

2.7 The Uprising of Deep Learning

Artificial Neural Networks (ANN) can trace their beginnings as far as the 1960's of last century. They are not by any means a "modern" technology. Then, in the 1990's they boomed again but as the new millennia started they went into oblivion once again. It was not until the early 2010's that ANNs started to blossom once more [53]. It was the beginning of a new era, and all the past shortcomings that kept ANN from truly succeeding were overcome, with the advent of affordable technology and decent amounts of information. The idea behind ANNs is that they can emulate the behavior of biological neurons, as in the human brain. The several parts of their inner mechanism such as *feed forward pass*, *backpropagation*, *loss functions* or *gradient optimization* could not be efficiently processed because of the lack of proper computational power and sufficient data to be trained on [65]. But as the 2010's decade started, *Graphics Processing Units* (GPUs) were put to new uses besides *gaming* and *multimedia processing*. GPUs have the ability to process with great speed and efficiency matrix operations, such as those needed in an ANN algorithm, thus a new application of GPUs was envisioned.

Nowadays GPUs are quite affordable, almost any researcher, academic or student can easily have access to proper GPU processors. Also, the CPU, RAM memory and storage needed to complement them are also very inexpensive. In addition, since the beginning of the massive access to the Internet in the mid 1990's, digital data banks have been geometrically growing ever since. For many years the use of such amounts of data was limited to the output of traditional *transaction-oriented information processing systems*. Then, *analytical processing applications* such as *business intelligence* or *data mining* started to conceive a new way to address great amounts of information to produce "knowledge", thus a new set of concepts started to flourish like *Big Data*, *Data Science* and of course *Deep Learning* (DL) [17].

As of now, the many everyday applications of DL include but are not limited to: image and video recognition for security usage, proper multi-lingual translation, voice recognition, biometric identification, intelligent assistants, automated customer service and automatic digital content production [4, 17]. It is clear the relevance of DL nowadays, and NLP is an area that continually benefits from research in it. In the AP field DL has yet to become a predominant trend, nonetheless the last couple of years have seen several studies that incorporate DL concepts. The reader will notice in next chapters, that the second proposal in this work addresses AP tasks by using DL ideas, outperforming SoA DL strategies in several international shared tasks.

2.8 Recurrence: an algorithm to remember

RNNs is a family of several architectures that are meant to tackle problems with a sequential nature. Take for instance the stock market or the weather prediction, these are considered *time series* type of problems [17]. Natural Language Processing (NLP) is also considered a sequential task, since most languages in the world are processed by humans as sequences of tokens (e.g., words, ideograms, pictograms). Regardless of the dialect, most languages are processed either as *left-to-right* or *right-to-left* progressions, which makes NLP a legitimate problem to address by the use of RNNs. The original RNN architecture has evolved drastically through the last decade, but the original idea still remains the same. Much like humans process sequential information, by "remembering" past details to predict the immediate future, RNNs also make use of different memory structures called cells for recalling important past facts, that might be relevant for the current piece of data being analyzed [17].

There are several recent successful stories of the use of RNNs. For example the work of Liu et al. in [66]. The authors of this research attempted a multi-topic sentiment categorization of emails using a *Bidirectional Long-short Term Memory* (BiLSTM) network. Their results showed a clear success of their approach. Another interesting example of the application of RNNs can be found in the work of Chow [67]. The author of this sui generis research focused on the prediction of auction prices of licence plates. In several countries the numbers, letters or even pictograms that appear in a licence plate can be chosen by the driver. In highly superstitious populations such as the Chinese, this is a paramount activity [67]. The author attempted the forecasting of the price using a NLP approach with the use of a RNN. By using 13 years of historical auction prices, the author was able to justify up to 80% of price fluctuations.

Although neither of the two approaches presented in this work makes direct use of RNNs, it is important to establish their importance in the NLP field, since until just a couple of years RNNs were the best choice to approach NLP endeavours. In addition, RNNs are the immediate antecedent of current SoA DL methodologies, such as the *transformer*, directly related with this thesis.

2.9 Attention Please!

Attention mechanisms have become very popular in the NLP community in the last couple of years. The pivotal paper *Attention is All you Need* [68] came as a game changer. Attention networks simplify RNNs, which in turn have produced the also novel and momentous architecture known as the *transformer*. What this type of architecture mostly do is compute the "importance" of the words in context of the neighboring terms, related to past and also future tokens, eliminating the need for complex memory cells like LSTMs or GRUs [69]. Most current research works done in the NLP field now make use of attention mechanisms, sometimes along with RNNs or even just as a transformer. For example recommender systems, that could vary from product suggestions to even hashtags proposals for images or microblogs. In the studies of Liu et al. and Li et al. in [70, 71], they make use of RNNs with attention mechanisms to guide the learning of the network over information of interest that can be relevant to the task at hand. In addition, another NLP field of interest is the analysis of quality in language, for example the quality of a given text could be assessed from the point of view of *semantics*, an *expert's opinion* or *popularity* among readers. The work in Luo et al. in [72] propose a RNN with an attention mechanism that offers a faster methodology to evaluate the quality of English texts. Then, there are cases of DL architectures that discard memory cells and make use of only attention mechanisms, this is the situation with the work of Liang et al. in [73]. The authors proposed an attention based network that addresses the rating of products through review information. Their approach devises a specific attention module to deal with the user's preferences and the features of the products they are reviewing. Also, attention mechanisms can be added to other type of networks, take the example of Rizzo et al. in [74]. The researchers approached the problem of automatic text generation with the help of an Adversarial Network, that implemented a self-attentive mechanism to improve the outcome. Finally in [75], González et al. used and encoder module derived from the *transformer* architecture to contextualize WEs trained over Twitter collected data for detecting irony in such social media platform. They attained very good results, achieving 1st and 2nd place in two international competitions. In deed "attention" in the context of NLP delivers outstanding results. Nonetheless, it might not be "all you need", as the lector will see in **Chapter** 5, *The Profiler* architecture proposed in this thesis demonstrates a combined effort of "attention" and latent extracted features, which delivers better performance than using just the former.

2.10 Merged algorithms or the origin of symmetry

Multitask Learning (MTL) is a way to enhance the generalization and possibly the prediction prowess of a neural network. It works by merging training samples from multiple tasks (targets). The sharing of these training instances guides the model towards learning to generalize better [17]. Even though MTL actually precedes DL for more than a decade, in the NLP field has recently become a feasible option to explore [76]. Take for instance the work of Akhtar et al. in [77]. In this research work the authors tackled simultaneously the tasks of *aspect sentiment classification* and *aspect term extraction*, since this problems are related (actually are co-dependent) [77]. The MTL approach employed by the authors fuses a BiLSTM network, a self attentive module and a CNN, achieving competitive results in three datasets in the English and Hindi languages. Likewise, the work of Lu et al. in [78] uses an autoencoder generative model to address two tasks: *learning text representations* and *sentiment analysis* of several Amazon reviews datasets. They hypothesize that both tasks are somewhat connected and the combined learning of them results in an improvement. Their experiments attained SoA results, outperforming methods based in single task learning. Moreover, the currently relevant phenomena of taxi transportation and/or ridesharing services such as Uber, can be optimized using a blend of DL and MTL approaches. That is the idea of Luo et al. in [79], where the authors of such research attempted the forecasting of the demand by examining *spatiotemporal* dependences to establish causality relationships among several traffic zones. This could help stabilize the demand in under and/or over supplied areas [79].

The premise of learning multiple "things" as seen in MTL, can be used from the stand point of features. Instead of learning to predict several targets, we can simultaneously learn different types of features in order to predict one target. That is the case of the *Wide Deep Learning* (WDL) architecture proposed by Heng-Tze Cheng et al. in [13]. They proposed a joint learning approach built from a wide and a deep component. Their objective was to merge the ability to memorize features of wide models (e.g. logistic regression) and the generalization prowess of deep models for more accurate recommender systems, such as those used for the Google Play store. Departing from ensemble approaches, the WDL method combines the output of both models using a weighted sum, then uses a common loss function for joint training, whereas ensemble models employ a separate training phase, only to combine their respective outputs in a voting strategy, even requiring more computational resources. A natural question can be posed from this "joint" architecture, such as what each component is learning. Nguyen et al. tried to answer this argument in [80], by identifying an emerging *block structure* that varies depending of the form of the network. Ultimately it can be presumed that this block, although constantly present, might represent different modes of learnt features depending on the task addressed. These concepts will be revisited in **Chapter 5**, since they were used to inspire the proposal of a novel DL architecture called *The Profiler*, useful in AP tasks.

Chapter 3

Theory Framework

In this chapter the reader will find the explanation of theoretical concepts, employed as scientific basis for the proposal of two novel methodologies, which are the central point of this thesis. This chapter will try to clarify the intended purpose of several well known Machine Learning (ML), mathematical and statistical methods and techniques, so the lector can fathom the ideas behind the proposed approaches.

3.1 Machine Learning and its types

ML is a sub type of the Artificial Intelligence (AI) field. Its main premise relies in producing algorithms that could learn from data. Opposed to traditional Computer Science (CS) programs, the behavior of a ML technique is not pre-programmed. ML tries to mimic the human way of learning. Either a person could learn by example (e.g. the colors, letters), or they can learn by analyzing non-categorized information by trying to find intrinsic patterns within data [17].

3.1.1 Supervised Machine Learning

This type of learning uses datasets comprised of samples formed by features, and a specific target. The latter could be presented in the form of a label or a value. When producing datasets for Supervised Learning (SL), it is important to make sure that a fair amount of samples are available, since ML algorithms perform better with a vast number of samples [65].

3.1.1.1 Classification

Depending on the target of the dataset samples, a ML problem could be posed as a *classification* task. This consists of trying to predict the category of a specimen (e.g. gender of a person, type of a flower) [65]. Fig. 3.1 depicts how a supervised classification task works.

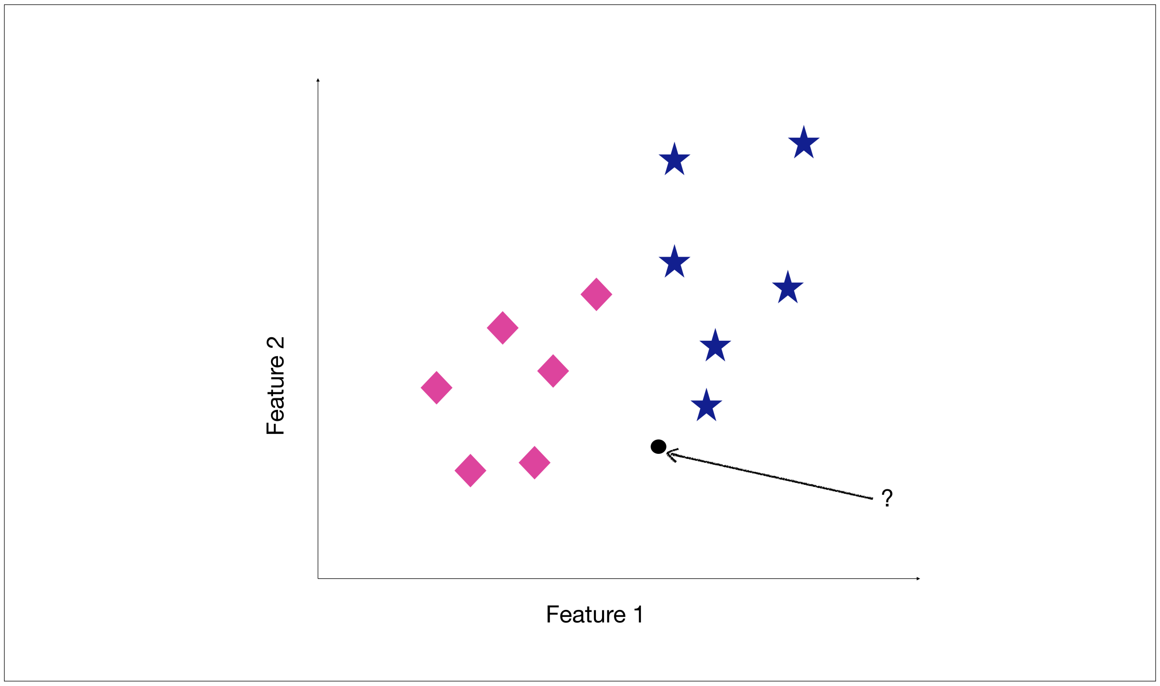


Fig. 3.1. Classification in two dimensions.

3.1.1.2 Regression

When the target is a value to predict, usually in the form of $tgt \in R = \{x \in R\}$. The task is said to be posed as *regression*. Many real life problems such as predicting the market price of a real state, time series or even the stock market fall into this category. Fig. 3.2 shows a typical regression problem in two dimensions [17].

More often than not, ML problems are tackle with multi-dimensional large datasets. In Fig. 3.1 and Fig. 3.2, for educational purposes only two dimensions and a few samples are depicted. In order to show geometrically a representation of data, several *dimension reduction* techniques can be used, such as *t-distributed Stochastic Neighbor Embedding* (t-SNE) and/or *Principal Component Analysis* (PCA) [69, 81].

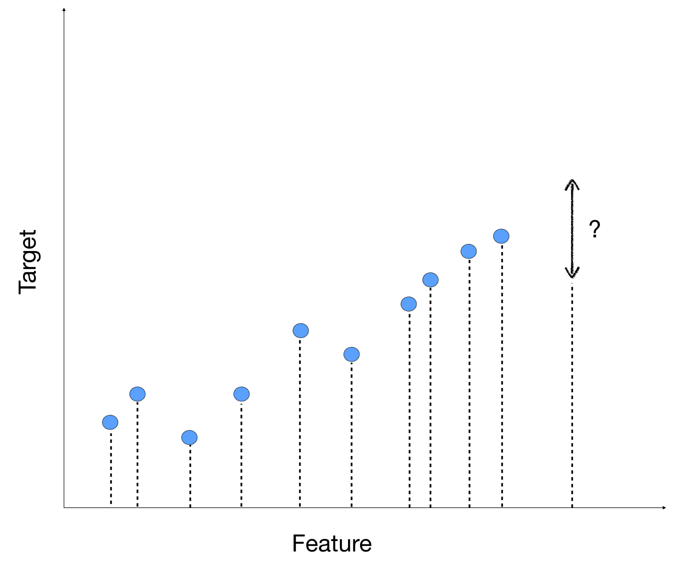


Fig. 3.2. Regression in two dimensions.

The Author Profiling (AP) problem tackled in this thesis is addressed as supervised learning, since all available datasets are labeled (categorized). Then most tasks are posed as a classification problem (e.g., gender or age prediction), and just a few ones as regression problems (e.g., personality traits).

3.1.2 Unsupervised Machine Learning

Most of the datasets available for ML algorithms are not labeled, since this is a costly task. Unsupervised learning (UL) deals with such information banks, frequently by finding underlying patterns [17]. Fig. 3.3 depicts how four different classes could cluster in a two dimensional space. The key of an unsupervised learning algorithm, is to find the pattern among the dataset points, so that it could either help a labeling sub task or even extract knowledge from it. UL techniques are mostly used for *clustering*, *anomaly detection*, *visualization* and *dimensionality reduction*. Among the most frequently employed UL algorithms we can find: *K-Means*, *Principal Component Analysis* (PCA) and *Distributed Stochastic Neighbor Embedding* (t-SNE) [53, 81].

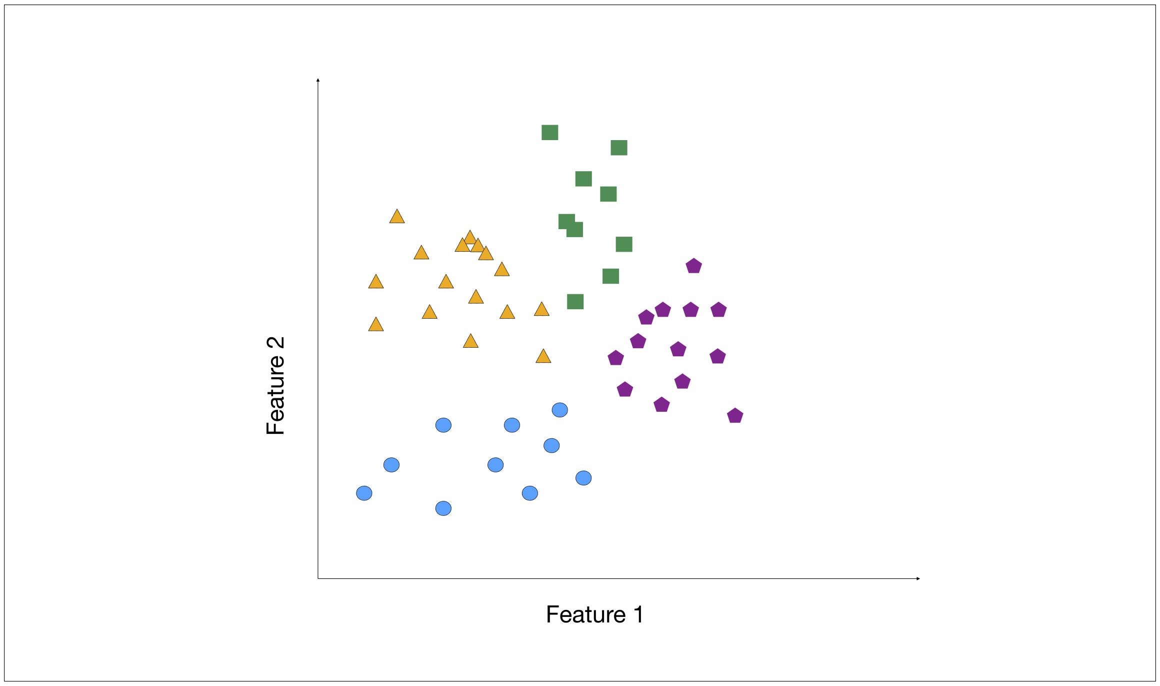


Fig. 3.3. Four different clusters in two dimensions.

3.2 Machine Learning and its different algorithms

Nowadays, ML techniques varies from traditional approaches to State of the Art (SoA) architectures. Even though Deep Learning (DL) has caught the main spotlight, traditional methodologies are still relevant. In the next few lines, the main ideas behind several traditional and modern ML algorithms are explained, all of which have been employed to some degree in the current thesis.

3.2.1 Support Vector Machines

Created in the 90's by Vladimir Vapnik et al., *Support Vector Machine* (SVM) is an algorithm with a strong mathematical support behind it. Its extraordinary efficacy in small to medium size problems (where the training samples are not huge) and feature complexity, makes it a favorite among the ML community. It can be employed to tackle linear and non-linear problems, and also it can be used in both *classification* and *regression* tasks [65]. Since SVM was one of the top predictor algorithms in the first proposal of this work, it is important to understand its mechanism, as it will make it easier to grasp the reasons behind its success.

SVMs are known to be very sensitive to non-scaled features, so it is recommended a pre-processing step of the data, which needs to include a form of scaling or normalization of the features [69]. Fig. 3.4 depicts a linear SVM classifier. Take notice in the circled samples that are precisely located very near the sides of the street, these are the *support vectors*.

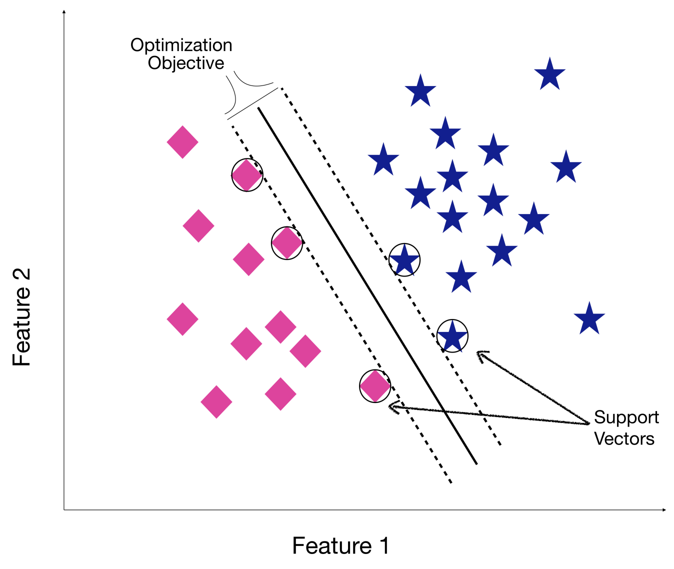


Fig. 3.4. A traditional linear SVM classifier with a *hard margin* classification approach.

3.2.1.1 Hard & Soft margin classification

When the task at hand is classification, the aim of the SVM algorithm is to find a “street” as broad as possible between the training samples of the dataset [69]. The solid line in Fig. 3.4 is known as the boundary between classes, and the objective is to position the dashed lines parallel and as far away as possible from the boundary, without allowing any sample to invade the street, since that is considered as a miss classification or error, this is known as *hard margin classification*. Because many problems are not as “ideal” as the one displayed in Fig. 3.4, a policy that allows *“a few”* samples to be misclassified in favour of a wider and more general solution would be preferred in some cases, this is called a *soft margin classification*. The hyperparameter C is used to adjust this margin, this variable allows to penalize the model so a suitable *trade-off* between *accuracy* an *generalization* can be achieved [69]. The bigger the penalization (C) the less of the samples that are allowed to enter the street. On the other hand, when the C value shrinks, more samples are allowed into the street, which decreases the accuracy in favor of a more general performance across different datasets [69]. Fig. 3.5 shows two models that allow a few samples to be miscategorized to favor a model that could generalize better in unseen data. The right plot allows fewer mistakes (larger C), increasing the accuracy of the model, whilst the left plot is less accurate but probably performs generally better among several datasets (smaller C).

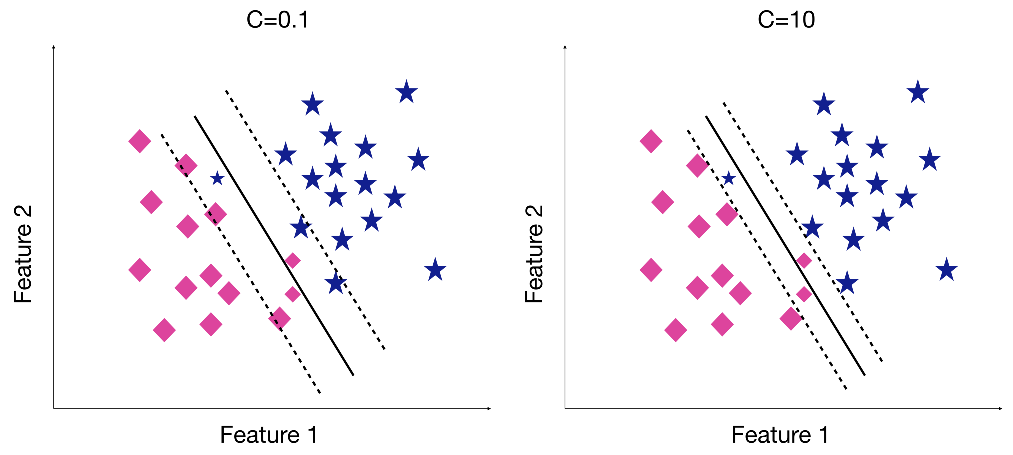


Fig. 3.5. Hyperparameter C penalizes the SVM model, so few (big C) or many (small C) errors are allowed inside the "street".

3.2.1.2 Non linearly separable data

The samples in Fig. 3.5, represent a clearly linear separable problem, thus a SVM classifier could perform generally well. But many other datasets are not as easily approachable. When this happens some transformation could be applied to the features, so the instances in the dataset could be separated [69]. Take for instance Fig.

3.6, we can see the same dataset in one and two dimension. In the left plot, the uni-dimensional version of the dataset is clearly non linearly separable, whilst the same dataset with a two-dimension projection can now be separated through the green line.

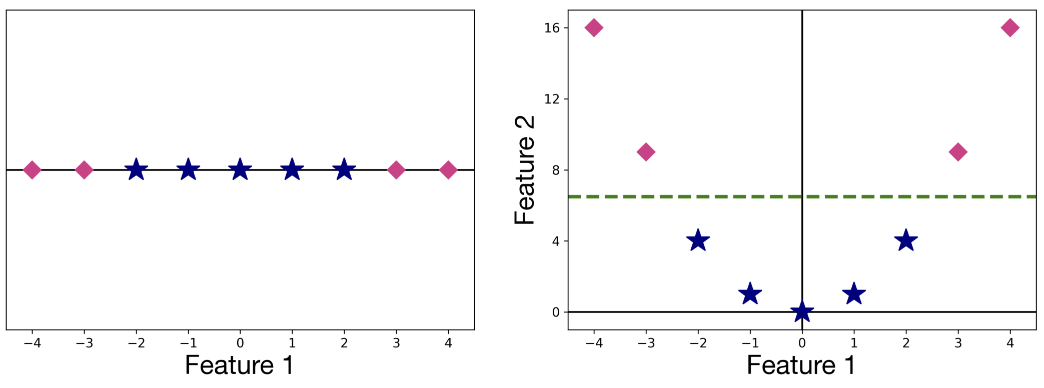


Fig. 3.6. A dataset projected in one dimension in left plot, and two dimensions on the right, using a quadratic function.

Other techniques that could render a higher dimension projection include *similarity functions* [69]. Take for instance left plot in Fig. 3.7, the uni-dimensional dataset explained before now can also be projected into a higher dimension space using *similarity functions*. The left plot in same figure shows similarity functions X_2 and X_3, established by two *landmarks* from the original dataset, where all points can now be extended into a new two-dimension space by calculating the *similarity* of each point to each function. Ideally there could be as much *landmarks* as points in the dataset, buy for explanatory purposes only two are chosen. The new features are computed using Eq. 3.1, which is a Gaussian radial basis function (Grbf). Where x is the point to be projected into a new dimension, ℓ is the *landmark* of the similarity function, and γ is a hyperparameter that regulates the width of the *similarity function* [69]. As shown in Fig. 3.7, using the Grbf function explained above, we can see the one-dimension dataset transformed into a two-dimension space, is now linearly separable.

$$\phi_\gamma(x, \ell) = exp\Big(-\gamma \|x - \ell\|^2 \Big) \tag{3.1}$$

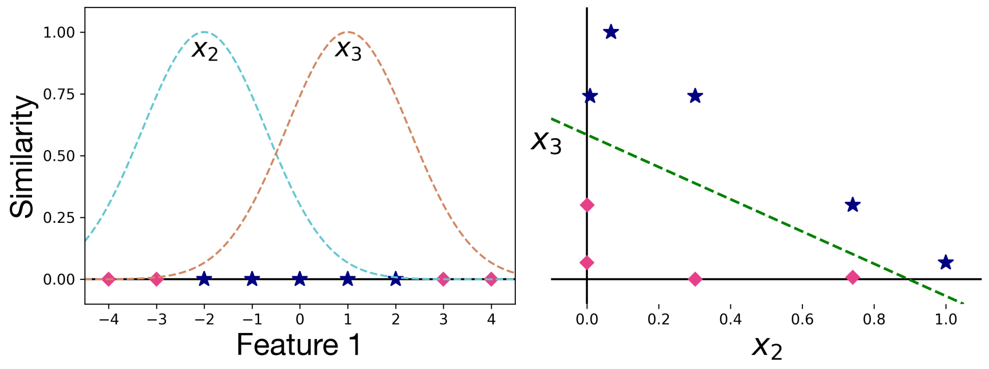


Fig. 3.7. Same one dimension dataset projected into a two dimension space by a Gaussian similarity function.

3.2.1.3 The kernel trick

As just explained before, when a dataset is expanded into more dimensions, the likelihood of now being linearly separable increases. But when dealing with huge datasets (as most of the times happens), the amount of computational resources to actually project data points into new dimensions is huge. This is why the real benefit of SVMs is to find the optimum "street" between classes in a n-dimensional space that could be separable by an hyperplane, with no additional computational cost for projecting into more dimensions [69]. The SVM algorithm can accomplish this using the *kernel trick*.

To better understand this let us consider a few mathematical concepts. If you would want to separate data points in a two-dimensional space, a line would be perfect to accomplish that. From analytical geometry we know that a line equation is represented in Eq. 3.2. Where m is the slope of the line, and b is the offset in the y axis.

$$y = mx + b \tag{3.2}$$

Eq. 3.2 works as a “separator” in a two-dimensional space, but this concept could be extrapolated to any n-dimensional space, where a decision function of the same dimensionality intersects with it, producing a hyperplane of $n-1$ dimensions, that serves as a boundary frontier for classification. Eq. 3.2 can be generalized for any n-dimensional space producing Eq. 3.3, where $\mathbf{W}^\top$ is a tensor of weights and X a tensor of features [69]. In Fig. 3.8 a two-dimension dataset and a decision function intersects in a one-dimension hyperplane (a line), which serves as the “street” (frontier) between two classes.

$$\mathbf{W}^\top X = 0 \tag{3.3}$$

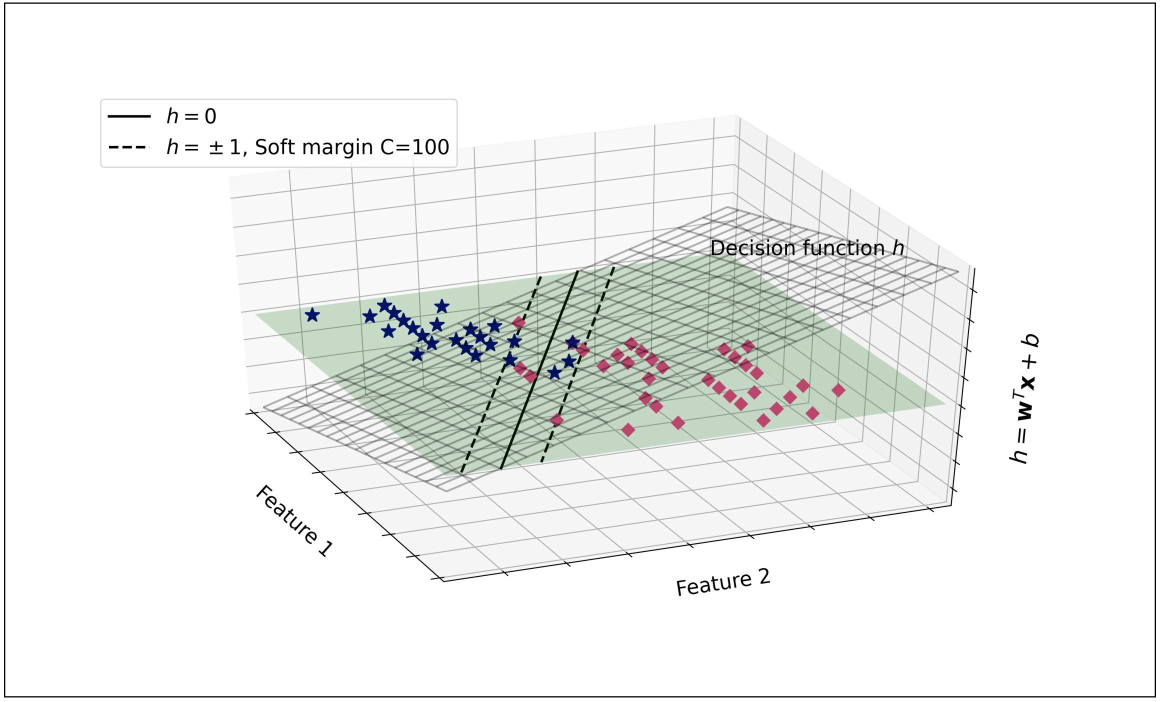


Fig. 3.8. A hyperplane and a decision function of n-dimensions, will intersect over a decision boundary of $n-1$ dimensions.

If we would want to produce a very accurate model, then the samples in the “street” need to be restricted. At the same time, it is useful to make the “street” as wide as possible, in favor of a model that generalizes the best. In deed is evident that if **W** increases, the "street" becomes narrower, and the opposite is true when **W** declines. This is why the model becomes an optimization problem where we need to minimize **W** [69]. Simultaneously we need a safe margin within said **W** (width of the "street"), that needs to be free of data points, that means that no errors are allowed inside. In Eq. 3.4 we can find the optimization problem for a hard classification strategy. Let us say that the width of the "street" is 1, then take special attention to t_i, which is a value that becomes 1 if the data point is above the boundary, and -1 otherwise. By doing this we ensure that no data point is allowed within 1 unit from the center of the "street" (decision boundary) [69].

$$\begin{aligned} \underset{w,b}{\text{minimize}} \quad & \frac{1}{2}\mathbf{W}^\top\mathbf{W} \\ \text{subject to} \quad & t_i\Big(\mathbf{W}^\top\mathbf{X}_i + b\Big) \geq 1,\ i = 1,\ldots,m \end{aligned} \tag{3.4}$$

The reason to minimize $\frac{1}{2}\mathbf{W}^\top\mathbf{W}$ (which is equal to $\frac{1}{2}\|\mathbf{W}^2\|$, as a result of matrix multiplication rules), instead of $\|\mathbf{W}\|$, is because it is has a useful differential [65, 69]. If a soft margin classification is needed, then a few additions to this problem could be made. In Eq. 3.5 we can now see a cost term for each data point that crosses the frontier, where ζ_i is a slack variable, that adjusts how much a data point is allowed to "enter" the "street", and C is the hyperparameter that penalizes the errors, that being said the higher the value of C the less samples are allowed into the "street" [65, 69].

$$\begin{aligned} \underset{w,b}{\text{minimize}} \quad & \frac{1}{2}\mathbf{W}^\top\mathbf{W} + C\sum_{1}^{m}\zeta_i \\ \text{subject to} \quad & t_i\Big(\mathbf{W}^\top\mathbf{X}_i + b\Big) \geq 1 - \zeta_i \ and\ \zeta_i \geq 0,\ i = 1,\ldots,m \end{aligned} \tag{3.5}$$

This optimization problem is convex and quadratic with linear constraints. To solve it, a technique known as *Quadratic Programming* is often employed [69]. Eqs. 3.4 and 3.5 work pretty well with linearly separable problems. Nonetheless when the dataset needs a higher dimension projection to become linearly separable, this *primal* "form"of the problem becomes computationally expensive. The concept of *duality* is very useful in mathematics, that implies that a *dual* form of the problem also exists. Under certain circumstances the solution to the *primal* and *dual* problems is the same, and SVM happens to meet this conditions. The advantage of solving the *dual* form is that it admits the use of the *kernel trick* [65].

To pass from the *primal* to the *dual* form of the problem, a method called *the Lagrange multipliers* is employed. Only this method exclusively allows equality constraints, that is why the *generalized Lagrangian equation* is used instead, which also accepts inequality constraints. When the Karush–Kuhn–Tucker conditions are met (SVM also complies with these conditions), the constraints are included in the optimization equation, multiplied by an α value (*Lagrange multiplier*) [65, 69]. Eq. 3.6 shows this *generalized Lagrangian* for the SVM problem.

$$\begin{aligned} &\mathcal{L}\big(W, b, \alpha\big) = \frac{1}{2}\mathbf{W}^{\top}\mathbf{W} - \sum_{1}^{m} \alpha_i \Big(t_i \Big(\mathbf{W}^{\top}\mathbf{X}_i + b\Big) - 1\Big) \\ &\alpha_i \geq 0,\ i = 1, \ldots, m \end{aligned} \tag{3.6}$$

If the partial derivatives of this *generalized Lagrangian* with regard to W and b, are equal to 0 (to find the stationary points), then the results for W and b can be introduced into Eq. 3.6, and with a little algebraic magic we now have Eq. 3.7.

$$\begin{aligned} &\underset{\alpha}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j t_i t_j \mathbf{X}_i^{\top}\mathbf{X}_j - \sum_{i=1}^{m} \alpha_i \\ &\text{subject to} \quad \alpha_i \geq 0,\ i = 1, \ldots, m \end{aligned} \tag{3.7}$$

Now Eq. 3.7 is also the *dual* form of the problem, if we can find a solution that minimizes this form, it will be also a solution for the *primal* form. Pay attention to α, t and X, for the *dual* problem can implement them as *matrices*, which allows for the *kernel trick*. Now *Quadratic Programming* can be used to solve the *dual* problem [69]. Once the optimal $\hat{\alpha}$ is found, now the optimal $\widehat{W}$ can also be computed with Eq. 3.8.

$$\widehat{W} = \sum_{i=1}^{m} \hat{\alpha}_i t_i \mathbf{X}_i,\ \hat{\alpha}_i > 0 \tag{3.8}$$

Then, the *kernel trick* consists in finding a transformation of the *features* in the dataset, without the computational cost of actually performing the projection [65, 69]. Let us say that exists a ϕ function that is responsible for the projection. Then in Eq. 3.7 the dot (also known as inner) product $\mathbf{X}^{\top}\mathbf{X}$ would have to be in the form of $\phi(\mathbf{X})^{\top}\phi(\mathbf{X})$, which depending in the transformation could mean a high computational cost due to the increased number of features; for example a Grbf transformation could render an infinite number of features, which ultimately is impossible to compute. But there is an algebraic equivalence that says that $\phi(\mathbf{X})^\top \phi(\mathbf{X}) = \left(\mathbf{X}^\top \mathbf{X}\right)^2$, thus we can compute the dot product of vectors $\mathbf{X}^\top and\ \mathbf{X}$ without having to perform, not even know what ϕ actually is. $K = \mathbf{X}_i^\top \mathbf{X}_j$ in Eq. 3.7 is also known as the *Gramian* matrix, which is the dot product of the original features, and can be used as a lookup table [69].

Finally, the *training* phase of a *SVM* algorithm consists in solving the optimization problem discussed before. Then all the training instances that verify Eq. 3.9 are known as *support vectors*. Next, the remaining training instances can be disposed of, because only the *support vectors* are needed [69].

$$t_i\left(\mathbf{W}^\top \mathbf{X_i} + b\right) = 1 \tag{3.9}$$

Knowing the *support vectors*, optimal offset $\hat{b}$ of the "street" can also be computed with Eq. 3.10. The reason to compute the *mean* of the *support vectors* is to find a more stable value [65].

$$\hat{b} = \frac{1}{n_s} \sum_{i=1}^{m} \left(t_i - \widehat{\mathbf{W}}^\top \mathbf{X}_i\right) \tag{3.10}$$

Then, the *testing* stage consists of making predictions with decision function 3.11. Notice that $K\left(X_i, X_n\right)$ is the dot product between the new test data point to be classified and the *support vectors* X_i, since remaining training instances were discarded, as only *support vectors* are needed to perform the classification task. This also makes prediction even faster, as a result of a smaller dot product [65].

$$\begin{aligned} & h_{\widehat{W},\hat{b}}\left(\phi\left(X_n\right)\right) = \sum_{i=1}^{m} \hat{\alpha}_i t_i K\left(X_i, X_n\right) + \hat{b} \\ & \hat{\alpha}_i > 0 \end{aligned} \tag{3.11}$$

3.2.1.4 Regression with SVMs

Regression problems can also be solved using a SVM, the trick is to invert the objective function. In this case the aim is to fit as many training instances in the "street" as possible. Those samples outside the "road" are the errors, and the *sup-*

port vectors remain the same [69]. Fig. 3.9 shows a *Support Vector Regressor*. The ϵ hyperparameter now controls the width of the "street".

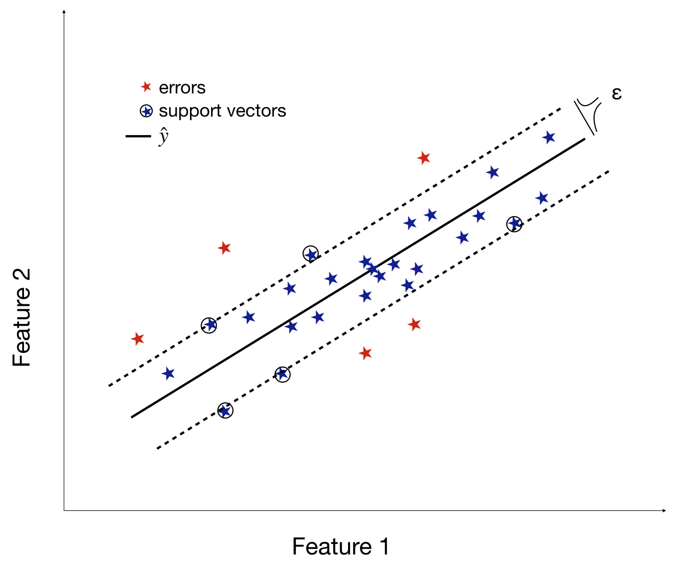


Fig. 3.9. The regression version of a *Support Vector Machine*, where training instances are fitted into the "street" as close as possible to the *target* value ($\hat{y}$ line).

3.2.2 Decision Trees

The *Decision Trees* (DT) algorithm is a very straightforward technique that deals with *classification* and *regression* problems as well. The main advantage relies in its structure being almost self explanatory, this means that the set of rules that devises are very easy to understand [65, 69]. The version of DT employed for this work is that implemented in scikit-learn ML library, which makes use of the *Classification and Regression Tree* (CART algorithm), that only produces binary trees [82]. Take for instance the very well known UCI wine dataset [83], a DT with a max_{depth}=2 is train over 2 of the 13 features available (for graphing purposes) (See Fig. 3.10). The root node is the *depth*=0, the next level *depth*=1 and the final tier *depth*=2. All nodes have four parameters and a rule, except the *leaves* nodes in last level, that have no rule since it is here where the final classification occurs. The two *features* selected out of the 13 available are *proline* (an amino acid present in wines) and *OD280/OD315 of diluted wines* (a measurement of protein content). The rule in node 0 divide the instances into those that have a *proline* level less or equal to the threshold value 755.0. The *gini* parameter is a measure of the "purity"of the node (more on this later). The *samples* parameter is the number of instances that are processed by the node, *value* is a vector of $n_{classes}$ that contains the information of how many instances of each class the *samples* are divided into. Finally the *class* parameter tell the category of the majority of instances in such node.

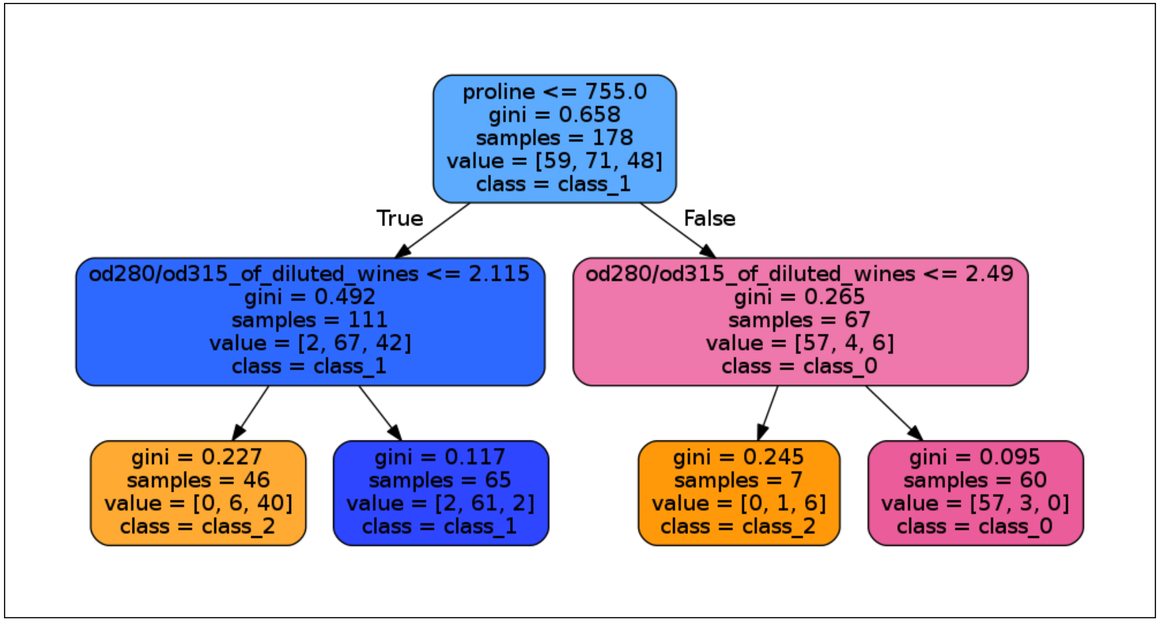


Fig. 3.10. A *Decision Tree* of the wine dataset.

The DT algorithm works by optimizing (maximizing) the "purity" of the partition established by the *gini* parameter. The purity of a partition is determined by how many classes of the minority co-exists within the majority class, that being said the less of the minority classes the more "pure" the partition is. A partition with 0 instances of minority classes is considered totally "pure". There are several ways to compute the "purity" of a partition, two of the most useful are *Gini impurity* and *Entropy*, the latter is borrowed from the concept of entropy in *information theory* [65, 69]. Eqs. 3.12 and 3.13 are used to determine the level of "impurity" of each partition.

$$G_{node} = 1 - \sum_{i=1}^{n\_classes} \left(\frac{value_i}{samples}\right)^2 \tag{3.12}$$

$$E_{node} = -\sum_{i=1}^{n\_classes} \frac{value_i}{samples} \, log_2\left(\frac{value_i}{samples}\right) \tag{3.13}$$

DTs make possible to estimate the probability of a sample of belonging to a specific class, which could not be computed with SVMs. Furthermore, the decision boundaries obtained when minimizing the "impurity" of partitions tend to appear orthogonal. This is the reason that often times a pre-processing stage such as *Principal Component Analysis* (PCA) is used to "align" the data, since PCA computes the components in a descending order of variability, and also are orthogonal between them [69]. In Fig. 3.11 the decision boundaries attained with the DT explained in Fig. 3.10 are set to a max_{depth}=2. It is important to regularize the model, because if no *max_depth* hyperparameter is defined, the DT will continue to divide until the error is the closest to 0 or it is impossible to continue, which will conclusively overfit the model.

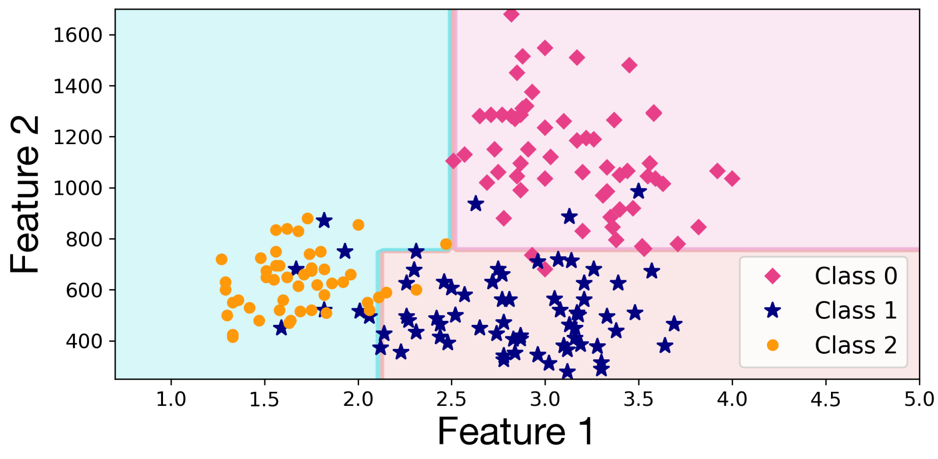


Fig. 3.11. The boundaries of a *Decision Tree* of the wine dataset, with depth=2.

3.2.2.1 Regression with DTs

As with SVMs, the DT algorithm can also be applied to solve *regression* problems. The mechanics is quite similar to the *classification* task. Instead of predicting a class, now a DT needs to forecast a value, only this is achieved by computing the *mean* target value of all instances in a partition. Also, instead of minimizing the “impurity”, an *error* metric is now optimized, for example the *Mean Square Error* (MSE), between the target value $\hat{y}$ of a node and the *average* target values of all instances in the partition. Just like with *classification*, if a DT regressor is not regularized, it most likely will overfit [69].

3.2.2.2 Random Forest

As well as SVM, *Random Forest* (RF) also performed admirably in the first proposed methodology of this thesis. RF is a collective method known as *ensemble learning* that demonstrates two heads are better than one, this is also known as *crowd wisdom*. For example, let us have several *decision tree* classifiers, we can train them on a subset of the *train* dataset and then make predictions by a *voting rule*. The class that gets the most votes is the winner. This “democratization” of the decision has been proven effective before. Take for instance a set of *weak learners*, meaning *predictors* that are barely better that a toss coin. The democratic decision of such learners will be more precise that each one of them on their own. This can be explained by the theory of *large numbers* [69]. As an example imagine a humongous bowl of shiny red balls, then suppose we ask a very diverse (in age, gender and even education background) set of 20 persons, to guess the number of balls inside the bowl. The answers would differ from an underestimation to probably an overstated number. Nonetheless, if we average (*aggregate*) the results, this answer will be the most accurate one. The same is true for a group (ensemble) of classifiers. If we make a group of *decision trees*, train them on random subsets of the train data, and make a voting prediction, then we have ourselves a *Random Forest* (RF).

The reason why *crowd wisdom* works is because of the diversity of the predictors. For an ensemble method to work, the individual classifiers need to be diverse, meaning that the errors they make during prediction do not correlate between them (which could happen if they are trained over the same dataset). This can be solved by aggregating predictors that are very different in their inner structure, for example a SVM, a DT and an *Artificial Neural Network* (ANN) [69]. But many times the computational resources during training of such diverse predictors can be prohibitive. A more homogeneous approach would be to use a unique type of estimator (DTs for example) and make them independent. This can be achieved by using random subsets of the *training* data in each DT (sampling). If this sampling is performed with replacement we call it *bagging* (bootstrap aggregating), otherwise it is called *pasting*. When the voting is made using the final classification choice of each DT, this is called *hard voting*. Some algorithms such as DTs allow to estimate the probability of each class for each testing instance (SVMs do not support this), then if we use this values instead of the final decision it is called *soft voting*, which often times deliver better accuracy[65, 69]. The *bagging* technique introduces a higher bias, but as the DTs end up being less correlated between them, the variance decreases, which results in higher accuracy [65].

3.2.2.3 Extra Trees

RFs differ from single DTs in the feature selection method for optimization. DT will find the optimum feature to begin the splitting. RFs randomly sample the feature set instead of exhaustively searching for the best feature. This favors greater diversity among single DTs of the RF, but they still need to find the optimum *threshold* in the domain of the selected feature. *Extremely Randomized Trees* or *Extra-Trees* (ET) not only will sample the feature set, but also will sample the *threshold* domain, resulting in an even faster, more diverse set of DTs, which will introduce even higher bias, and lower variance. In practice RFs and ETs perform very similarly. From a practical point of view, it is difficult to anticipate which will perform better in a particular task, this is why either (or even better: both) will deliver competitive results. As with DTs, RFs and ETs can also be implemented for *regression* tasks, the principle mainly remains the same [69]. ETs performed very similar to RFs in the first proposal of this thesis. Between both they attained more than 50% of the top results in 17 AP tasks. There were some other ML prediction algorithms that were tested, nonetheless only SVM, RF and ET are explained in this chapter, since they were the best performers.

3.2.3 Artificial Neural Networks

The second methodology proposed in this work is based on Artificial Neural Networks (ANN). ANNs are one of the greatest contributions of ML and AI in general. It mostly evokes the *human nervous system* and tries to emulate it by copying the functionality of its principal cells (neurons). An ANN is a structure that is comprised of an *input* layer, an *inner* structure (which could have several levels) and an *output* tier. It is a supervised learning method which correlates the features in a dataset with an output target, which can be a label or a value, so it can be used for either a *classification* or a *regression* task. When the ANN learns how to predict targets its "knowldege" is kept in its inner structure, in the form of weight values of the "artificial neurons", which are elements similar to human neurons that can be activated by a function, creating a "memory" of patterns. Ultimately we call this process "learning" [65, 17].

The main functionality of an ANN can be summarize in two steps. First, the *forward pass* processes the inputs and through activation functions decides which neurons to "activate". At the end of this step it is decided which class or value is forecasted. But several "epochs" of learning must pass so the learning can be perfectioned [53]. For each *forward pass* there must be a *backward pass*, which adjusts the prediction in case it was wrong. The *backward pass* of an ANN is crucial to adjust the inner layers so they can properly predict the target. Such *backward pass* starts with a *loss function* (L_ε) which gives information about how far away from the target the network currently is. The well known *Backpropagation* algorithm for ANNs, applies *differential calculus* to provide a solution to find the slope (or gradient) of L_ε with respect to the weights of the network. In other words, provides a way of dividing the error among the neurons in the inner layers [53]. An ANN can be viewed as a function that must be optimized, that is minimize L_ε. To do so, an optimizer must be employed: *Stochastic Gradient Descent* (SGD) is one of the most used algorithms to optimize ANNs. First of all, an optimizer solver must make use of the *Backpropagation* algorithm to compute the *gradient* of L_ε, then it minimizes the error as the training epochs go by. Among the most used *loss functions* we can find the *Mean Square Error*, *Mean Absolute Error* and the *Mean Bias Error* for *regression* problems, as well as *Hinge Loss* and *Cross Entropy Loss* for *classification* tasks. On the other hand, the most used *optimizers* include: *SGD*, *Adam*, *Adamax* and *RMSprop*. Fig. 3.12 depicts a classical Artificial Neural Network [65, 17, 53].

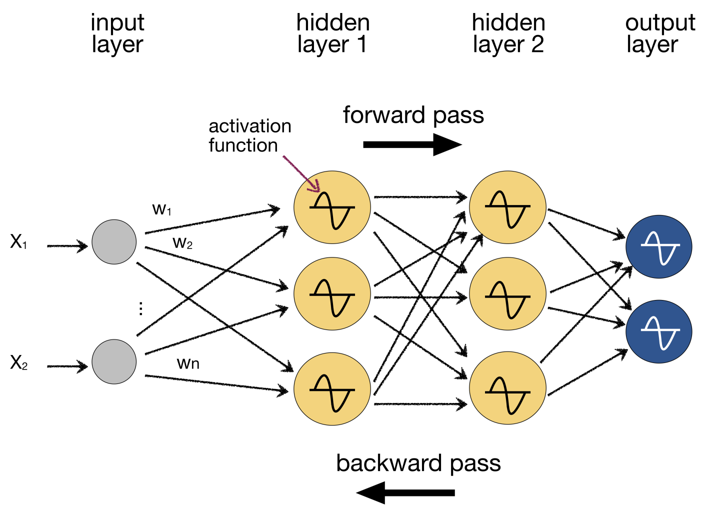


Fig. 3.12. A basic ANN architecture.

3.2.3.1 Recurrent Neural Networks

Although Recurrent Neural Networks (RNN) were not directly employed in this thesis, they are the immediate predecessor of the *transformer* architecture, which in deed inspired the second proposal of this work (explained in **Chapter** 5). RNNs are very similar to a simple ANN. However, RNNs address an issue ANNs often struggles with. When processing sequences, an ANN will not be able to "remember" old inputs that may have an influence on the current sample. *Time series* and *Natural Language Processing* (NLP) problems need to recall past inputs in order to better predict future targets [69]. A RNN can accomplish this by using a partial prediction to feed back the input in next step. So an input of a RNN will aggregate the output of last step (except in the beginning) with the current input vector, this stages are called *time steps* [17, 69]. RNNs can be used for several tasks depending on the architecture implemented. For instance, the partial output in each *time steps* could be used as a feature extraction technique, producing some type of embedding; an out of phase pair of stack RNNs are often used for translation jobs. The heart of a RNN can be explained by Eq. 3.14. Where vectors $\mathbf{W_x}$ and $\mathbf{W_y}$ are the weights of the current input and the last step output respectively, $\mathbf{X_t}$ and $\mathbf{Y_{t-1}}$ are the current input and last step output, ϕ is the output activation function, and b is the bias [17, 53].

$$Y_t = \phi\big(\mathbf{W_x}^\top \mathbf{X_t} + \mathbf{W_y}^\top \mathbf{Y_{t-1}} + b\big) \tag{3.14}$$

There are several RNN variants that try to optimize "how much" the network can remember. This is done by structures called *memory cells*. Depending on which type of *memory cell* the RNN uses, two sub-architectures are commonly implemented: *Long Short-Term Memory* (LSTM) and *Gated Recurrent Unit* (GRU). LSTMs and GRUs can be viewed as specialized RNNs, which can make some tuning to determine what past outputs must be more important when enriching the current input. This allows the network to remember some past outputs better than others, depending on which contributes the most [53]. Even though GRUs are newer and more sophisticated than LSTMs, the latter ones are more frequently used. In Fig. 3.13 we can see a RNN architecture, as explained mathematically by Eq. 3.14, unfolded through a time t.

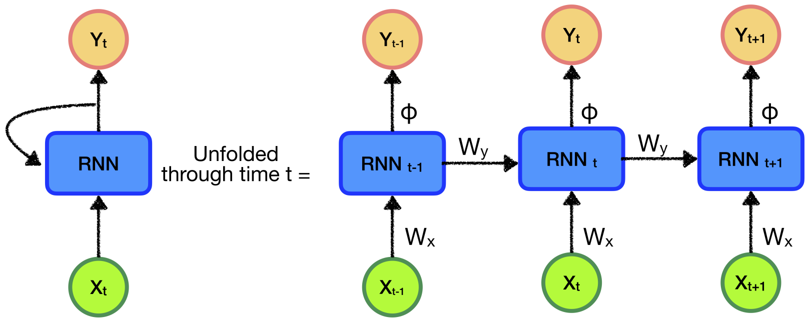


Fig. 3.13. A RNN architecture, where each current time step uses the last one to feed back the current input

3.2.3.2 The Attention mechanism & the Transformer architecture

The success of RNNs in tasks like *machine translation* is greatly documented. Nonetheless, this networks take huge computational resources. In 2017 a milestone paper called *"Attention is all you need"* [68] proposed a novel network architecture called the *transformer*, capable of paying attention to input sequences at different levels, without the need of any LSTM or GRU cells, which makes this architecture far more efficient to train. In Fig. 3.14 we can see the *transformer* network, explained in the context of a translation task. First we identify the *encoder* at the left and the *decoder at the right*. Both *encoder/decoder* are stacked N times. In the train step, the *decoder* receives the target sentence as well as the output of the *encoder* at the N stack to match the same level in the *decoder*. In the testing phase the *decoder* can not be fed the target sentence obviously, so the whole network is called token by token shifted one time step to feed the decoder with the previous output. Take special consideration in the *Multi-Head Attention* module, this layer encodes the relationship of the current word with the rest in the same sentence. Since the *Multi-Head Attention* layer is time-distributed (meaning that all words might be processed in parallel), the model has no information of the position of the words in the sentence, which is needed to know the context. This is why the *positional encodings* are fed at the beginning of the *encoder* and *decoder*. This embeddings have the positional information of the words. Notice the *Masked Multi-Head Attention* in the *decoder*, it is masked because only past words are allowed to be examined in this layer. The *Multi-Head Attention* layer is based on the *Scaled Dot-Product Attention*, which serves as a look-up dictionary for a sentence [69]. Take for instance the phrase "*Matt played a mean guitar last night*". The dictionary would have to be like this: Subject: "*Matt*"; Verb: "*played*"; Indicative: "*past tense*". Only the *encoder* does not come up with this straightforward dictionary. Instead it has a tensor representation of such look up table. This tensor dictionary is composed of three matrices, Q, K and V, representing a query, a key and a value matrix respectively. This dictionary works by computing a similarity score between the query and the key (the scaled-dot product), then it uses a *softmax* function to compute the value of the key. Since the *encoder* was able to encode different levels of information (e.g., the type of word, the indicative), then we also need several layers of *Multi-Head Attention*, this is why in each tier the several characteristics of each word are projected onto as many spaces, this is how the *trans-*

former is able to pay a more productive attention to a text sentence. The merit of a *transformer* is that outperforms RNNs by "seeing" the whole document at once, without the need of sequential processing and specialized recall modules (e.g., LSTM, GRU), which translates in a more efficient training stage (i.e., parallelized learning) [68, 69]. Eq. 3.15 explains the *scaled dot-product* that allows the multi attention span of the *transformer*.

$$Attention(Q, K, V) = softmax\Big(\frac{QK^{\top}}{\sqrt{d_{keys}}}\Big)V \tag{3.15}$$

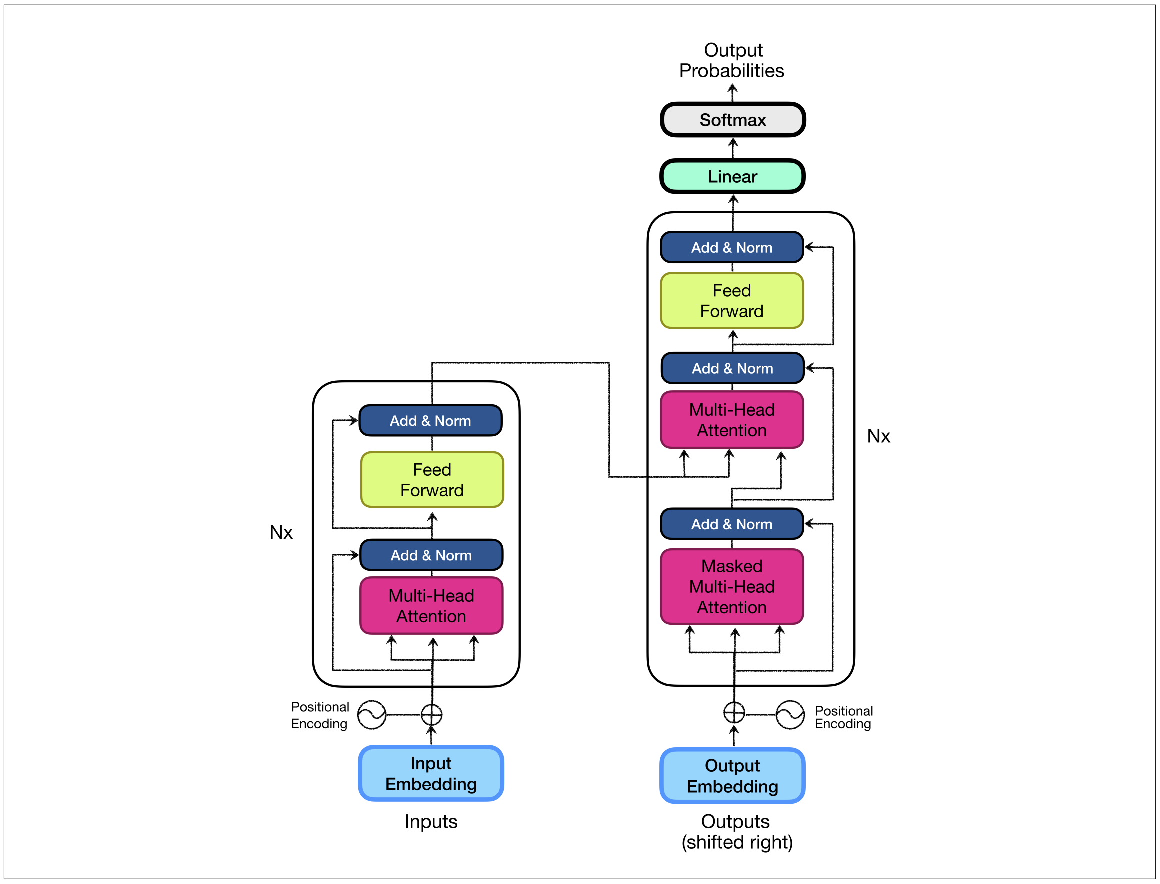


Fig. 3.14. The novel transformer architecture.

3.3 Document classification

The main objective behind both proposed methodologies of this work for tackling AP tasks, can be summarized as a *Document Classification* (DC) task. DC is one of the main and most approached problems in NLP. It encompasses a broad set of activities that include areas like *library, information and computer sciences.* From the perspective of *computer sciences*, DC aims to automatically categorize digital documents, which might include audio, video and text. *Digital textual documents* are of particular interest for NLP endeavours. It involves a vast number of techniques such as *machine learning algorithms* and *statistical value extraction* [53].

3.3.1 Textual representations & feature engineering

For DC or AP tasks, one of the main and current issues is *Textual Representations* (TR). Several methods have been tried over the years with satisfactory results. Nonetheless, the ever-changing scene of ML and NLP applications, often require this techniques to keep evolving. This is the reason why *word* and *paragraph* encoding methods have emerged massively in the last few years [17]. What are now considered "traditional" approaches include *one-hot encoding* and *Bag of Words* (BoW) methods; this techniques address TR from *vocabulary* and *statistics* point of views respectively. *One-hot encodings* produce a vocabulary dictionary with a numeric *id* for each word, then for representing sentences, it generates a vector of n dimensions (being n the number of words in the sentence) with the proper word id in the corresponding vector position. To represent single words, a *one-hot encoding* disperse vector will be created, with m dimensions (being m the number of words in the vocabulary), and with a number 1 in the position of the corresponding word, and zeros in the rest of the positions (disperse vectors) [53]. Fig. 3.15 shows a sentence encoded as a *one-hot encoding* vector.

3.3.1.1 term frequency–inverse document frequency (tf-idf)

The *term frequency–inverse document frequency* (*tf-idf*) is a numerical statistic often employed in NLP tasks (e.g., information retrieval) as a term weighting value[53]. This value can determine the importance of a word in a set of documents. The rationale behind this number is that a word's "importance" is related to the frequency

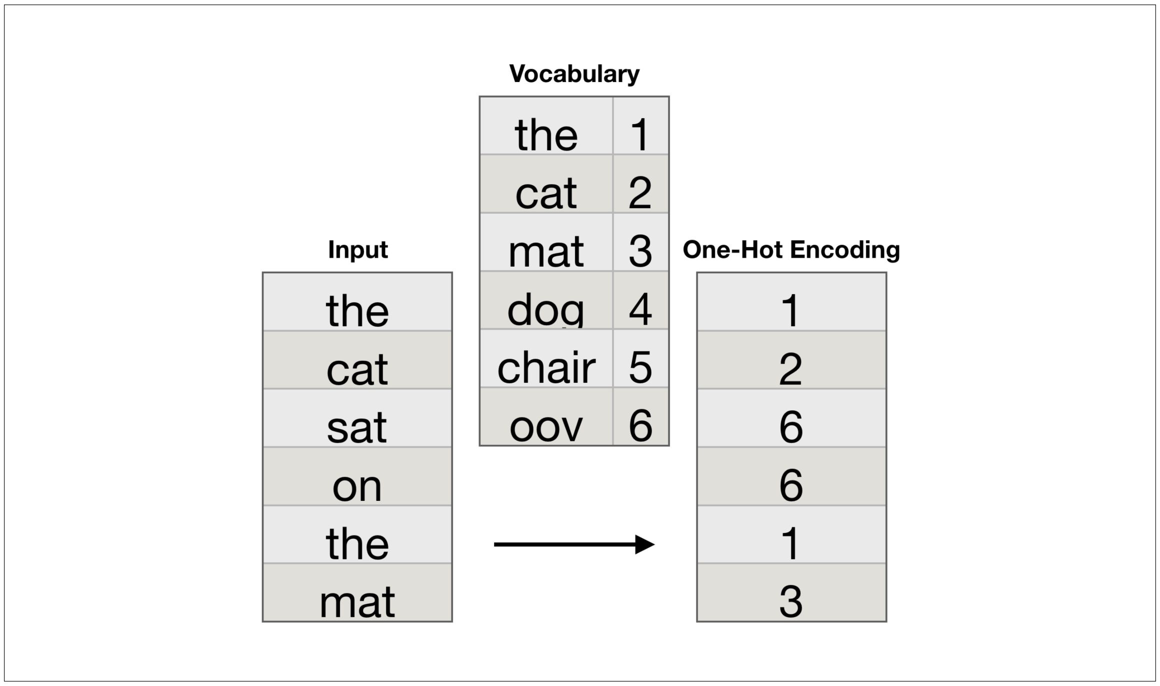


Fig. 3.15. The One-Hot Encoding method allows a very straightforward representation of text. Words that do not belong in the vocabulary are identified by a *out of vocabulary* (oov) token.

it appears in documents. However, a simple count of the number of times a term appears can be deceiving. Take for instance *stop words*, which are terms that are very frequently used but add no significant value towards the characterization of a document [53]. The set of *stop words* can vary between languages, examples of this words are *articles* and some *prepositions*. If a *stop word*'s frequency is taken into account, it can falsely magnify its importance. The *tf-idf* statistic not only considers the frequency of a term, but also offsets this value by decreasing its amount when a word is used frequently in most of the documents, which shows a non-profit behavior. On the other hand, if a word appears frequently in just a few documents, then probably its importance is far higher [53]. Eq. 3.16 demonstrates how to compute a *tf-idf* value for a single word. It is worth noticing that the *idf* value for each word (which offsets the frequency), is globally computed for the whole dataset. Observe that both terms (*tf* and *idf*) are computed in a logarithmic scale, due to the natural sparsity of the counting task. The *tf-idf* was a very important value used for computing weighting schemes in the first proposed method of this work (more on this in *Chapter* 4).

$$\text{tf-idf}(w_n) = (1 + \log \text{tf}(w_n)) \cdot \log \frac{N}{\text{df}} \tag{3.16}$$

3.3.2 Distributed representations

Circa 2013, *Word Embeddings* (WE) made quite an entrance in the NLP scene. This novel technique encode words by understanding their meaning in the text, and by projecting its connotation into a n-dimensional space, where terms with similar meaning tend to cluster together. WEs have greatly revolutionized the NLP field by adding *significance* to TRs. WEs are considered *distributed representations* because opposed to *one-hot encoding* vectors which are disperse, (can have as many dimensions as the length of the vocabulary) they can “distribute” the syntactic and/or semantic information in fewer dimensions [17]. WEs were the foundation of both approaches presented in this work.

3.3.2.1 Non-contextual Word Embeddings

Even though WEs encode the meaning of the word by predicting the “context” (surrounding terms), they are considered *non-contextual*. This is because only a single embedding will be create for each term. For example, if the term “Queen” appears in the dataset, only one WE will be produced for this term, even if the word is used is several contexts across the dataset, like “Queen” a title of nobility, or “Queen” the name of a rock band. This is known in linguistics as polysemy, or the coexisting different meanings for the same word [53].

word2vec

One of the most popular WE technique, that actually has paved the road for better improved approaches is *word2vec* (w2v). Mikolov et al. [8] proposed this algorithm as a clever way to encode meaning in a distributed representation; w2v works by using a “shallow” neural network that process *one-hot encoded* words from a textual dataset. There are two variants of the w2v algorithm: *Continuous Bag of Words* (CBOW) and *Skipgram*. The CBOW alternative is a network that tries to predict a word, given a fixed number of neighboring words. On the other hand, *Skipgram* aims the opposite, to predict the context given a word. Only the *Skipgram* variant is explained, since it is the most fruitful approach. Fig. 3.16 shows the main policy of both variants of the w2v algorithm.

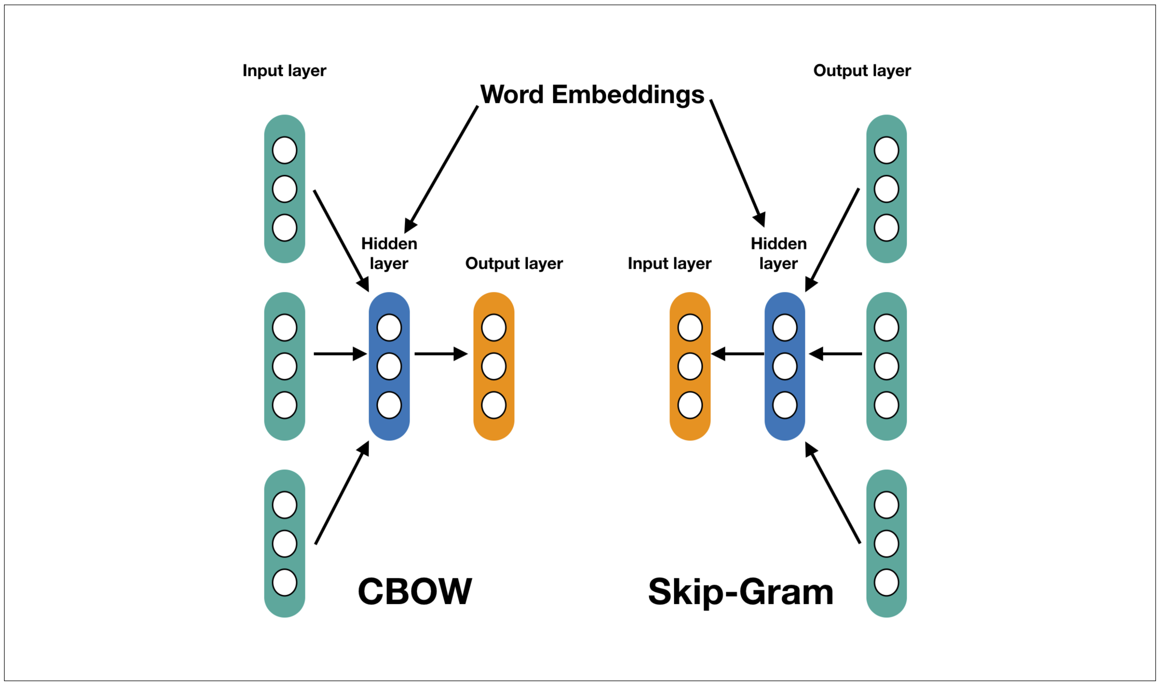


Fig. 3.16. The main architecture of the word2vec algorithm with its two variants: skipgram & COBOW.

Skipgram is a simple neural network that is fed with *one-hot encoded* words in the input layer. Then a single inner layer (hence the “shallow” motto) of neurons will record the weights of the learning process. The aim of the network is to make a probabilistic prediction of the surrounding words of the input term. The number of neighboring words is determined by the hyperparameter *window*. Empirically it has been documented that small window values will encode a more syntactic meaning of the term, whilst a larger window will produce a more semantic encoding of the word [8]. But where are the WEs? The task of forecasting the neighbor terms of the input word is actually a “fake” task. This means that what we are really looking for is the weights in the inner layer at the end of the training phase, these are the WEs. The number of neurons in the inner layer is actually the dimensions of the WEs [8]. Also empirically, WE dimensions of 100, 300 and 500 have been documented to produce useful embeddings. Fewer dimensions than 100 are usually not as beneficial, whilst dimensions over 500, will not deliver significant better performance, but will consume huge computational resources. *Skipgram* uses several "tricks" to accomplish its purpose in a relatively fast computational time, such as hierarchical softmax and/or negative sampling. Also it is worth mentioning that no traditional activation functions are used in the inner layer, instead, transformation matrices are used to encode the inputs into the inner layers, and then these into output vectors [8, 53]. Fig. 3.17 depicts the *Skipgram* variant of the w2v algorithm.

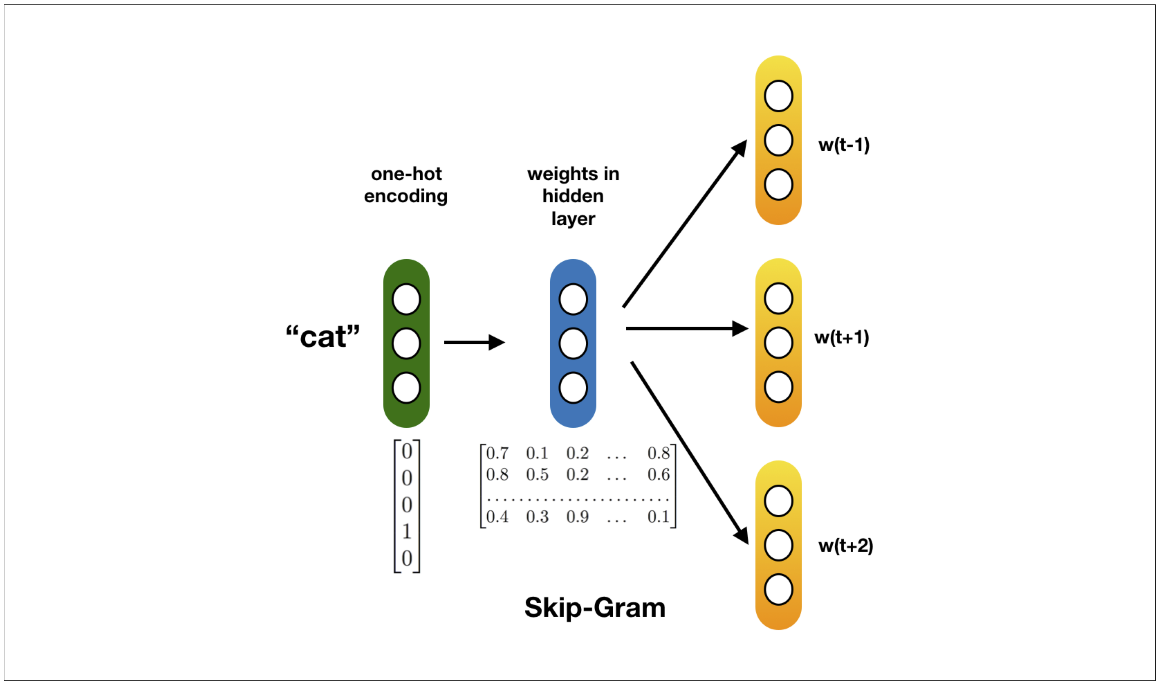


Fig. 3.17. The aim of the Skipgram architecture is to predict the context of a word, as a fake task to produce word embeddings.

fastText

Ever since w2v became the prevalent WE method, several word embedding techniques have derived from it, such as *Global Vectors for Word Representation* (GloVe) and fastText [84, 85]. The latter is an algorithm that pretty much works like the w2v *Skipgram* variant. The difference relies on the fact that the prediction is preformed at character level instead of words. This nuance departure from the w2v algorithm produces embeddings even for *out-of-vocabulary* (oov) terms, which are specially useful for datasets with many terms of this type, such as those produced from social media outlets [85, 53].

3.3.2.2 Contextual Word Embeddings

To address polysemy, among other shortcomings of WEs, 2019 saw an increase in embedding techniques, such as *Language Models* (LM). In this new generation of encoding methods the goal is to produce more advanced embeddings that deliver not only word representations (different vectors for the word “Queen” according to its contextual meaning), but even paragraph or whole document encodings. Up until LMs appeared, the way to produce TRs (embeddings) for paragraphs or documents, that is *Document Embeddings* (DE), usually an aggregate approach was used: averaging the WEs of the terms in a document [54]. Even though this method has delivered very competitive results, more specialized and successful aggregate procedures have been devised, such as weighted averaging based on the characteristics of the task or the dataset [86]. Even so, the quest for far-reaching methodologies continues. This is why LMs are now the SoA approach to perform tasks including but not limited to: producing WEs and DEs, translation tasks, question answering and natural language understanding [53].

Bidirectional Encoder Representations from Transformers (BERT)

A milestone paper for the NLP community was published in 2019 called “*BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*” [64]. BERT is a very potent LM that is based on the previously explained *transformer* architecture. To better understand how BERT works let us consider other popular LM: ELMo (Embeddings from Language Models) [21]. ELMo uses a LSTM network with two staked directional layers, which process input sequences in the first layer from left to right, and then in the second layer from right to left, finally concatenating both outputs for the output layer. On the other hand, BERT is based on the *transformer* architecture, and because it is a LM only the *encoder* part in needed. Departing from ELMo, BERT “sees” the whole sentence at once, thus making it a fully bi-directional model, but precisely because of this, it needs the positional embeddings explained before in the *transformer* network [64].

To train every neural network, a task must be defined, in this case BERT uses two prediction objectives: *Masked language modeling* (Mask LM) and *Next Sentence*

Prediction (NSP). The aim of Mask LM is to learn to predict *tokens* by analyzing the surrounding words, 15% of each sequence is "masked" with the [MASK] token. For NSP, the network is fed with pairs of sentences, where 50% of them are truly contiguous sequences and the other 50% are selected randomly, then the network must learn to differentiate actually adjacent sentences. Both prediction tasks are learned simultaneously with the aim of minimizing the combined loss function of the two approaches [64].

BERT pre-trained weights are publicly available, since it takes large amounts of time and computational resources to train. Then it can be fine-tuned to perform more specialized tasks [87]. This means that an additional output layer can be added to match the task that needs to be addressed, and only this last layer is trained (freezing the pre-trained weights), which makes the job less computationally heavy. In addition, when fine-tuning BERT also the option to retrain the inner layers is available, which makes small changes to the pre-trained weights. A wide impact of fine-tuning is debatable, for some problems might work well, but some other tasks can be less benefited from it, so the use of a BERT pre-trained model as it is can perform just as well [88].

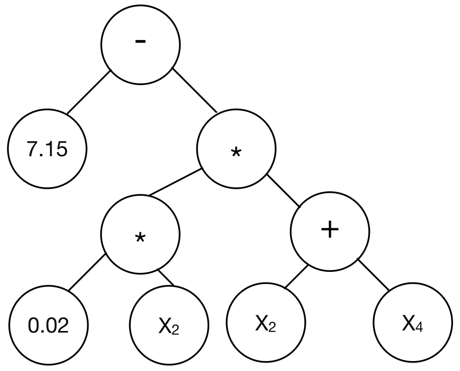
3.4 Evolutionary computation

The field of *Evolutionary Computation* (EC) is as sub area of *Artificial intelligence* which imitates the evolutionary process that takes place in nature, for example in most animal species. EC is mainly composed by several algorithms with an stochastic nature used for optimization purposes. Among the most well known members, *Genetic Algorithms* (GA) and *Genetic Programming* (GP) stand out [65]. GP was a crucial component for devising term weighting-schemes, which is the main contribution of the first proposed approach discussed in **Chapter** 4.

3.4.1 Genetic Programming

GP can be viewed as a natural extension of GAs. In the late 80's and early 90's of last century, John Koza invented and popularized this new kind of algorithm. The objective of GP is to evolve computational programs, Koza has referred to GP as an "invention machine", capable of creating computer programs on its own [89]. For regression problems, features can be posed as a set of mathematical equations that are structured as trees, which can be evolved through mutation and crossover operations, until these mathematical formulae are capable of reaching a target solution, by minimizing the error. Then, GP is fed with a set of mathematical operators (e.g., *, /, +, -, sin, cos, log), constants and terminals. The features of a dataset for GP are commonly used as terminals (variables), and an initial population of candidate equations is created randomly as trees [65]. See Fig. 3.18, this formula is a candidate specimen to solve a target value, whether this equation is fit enough, will be determine by a fitness function, which measures the error of the sample.

From the beginning of the procedure, GP's hyperparameters are defined, these include the number of *generations* the algorithm will run until, the number of individuals in the initial *population*, the probabilities for the *mutation* and *crossover* operations (usually 90% for *crossover* and 10% for *mutation*), the set of mathematical operators (*function set*), the number of *constants* and the *tournament size*. This last hyperparameter is used to establish how the best members of each epoch will participate for *mutation* or *crossover* operations to populate the next generation [90].



$$y = 7.152 - 0.02458\,\mathrm{x}_2\ (\mathrm{x}_2 + \mathrm{x}_4)$$

$$y = 6.9308$$

Fig. 3.18. An example of a Genetic Programming equation tree.

Performing *mutation* or *crossover* operations on a tree equation, consists in replacing certain branches of the tree, either by a random substitution or by swapping branches between two trees [65]. For example see Fig. 3.19, a *mutation* operation is performed on the candidate tree by changing one of its terminal branches with a new randomly produced substitute.

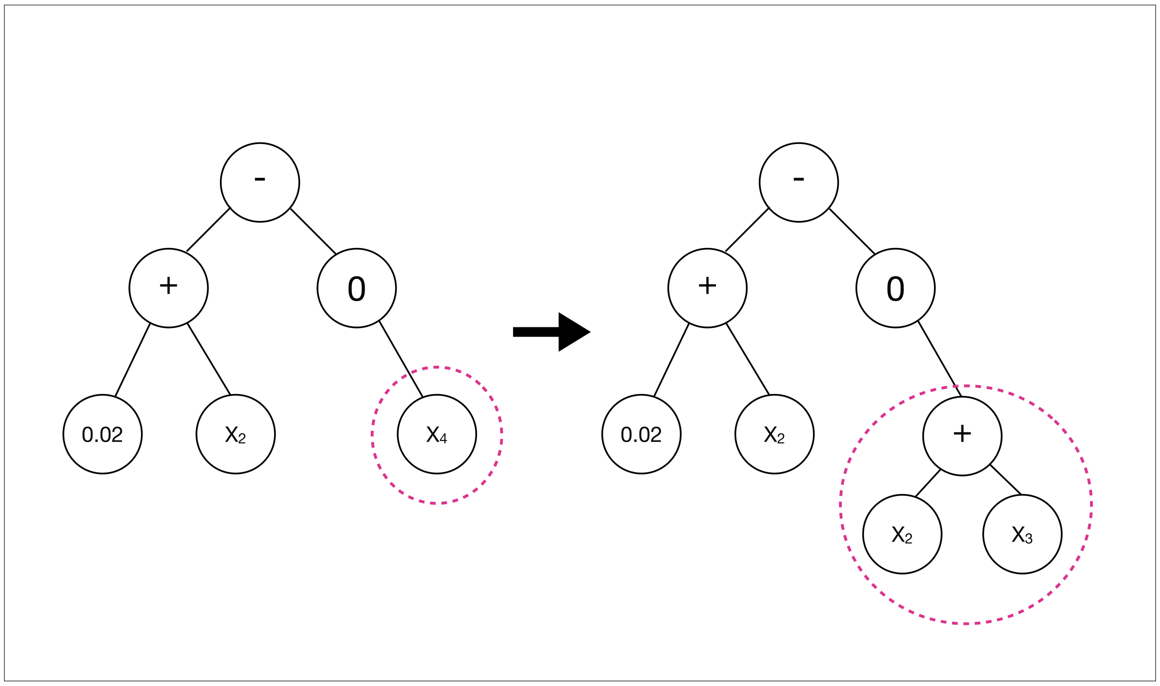


Fig. 3.19. The mutation operator performed in a tree in a GP population.

For a *crossover* operation see Fig. 3.20, two of the most "fit"members of the current population are selected (by probability) to participate in this operation. The most right leaves of each tree will be swapped between trees, thus creating a new pair of new specimens, that are theoretically more ''evolved", meaning are more fit to predict the target value [65].

Finally, a common "glitch" found in a GP process is called *bloating*. It consists in producing huge equations (trees) by continuously creating taller trees (large number of levels), that actually add no significant reduction in error, thus these equations, even if effective, are in no way efficient. Various techniques to deal with *bloating* are commonly introduced in the several implementations of GP available. Either by limiting the height of the trees or by early stopping strategies, bloating can be adequately dealt with [90]. Algo. 1 shows a general pseudocode for a standard GP algorithm.

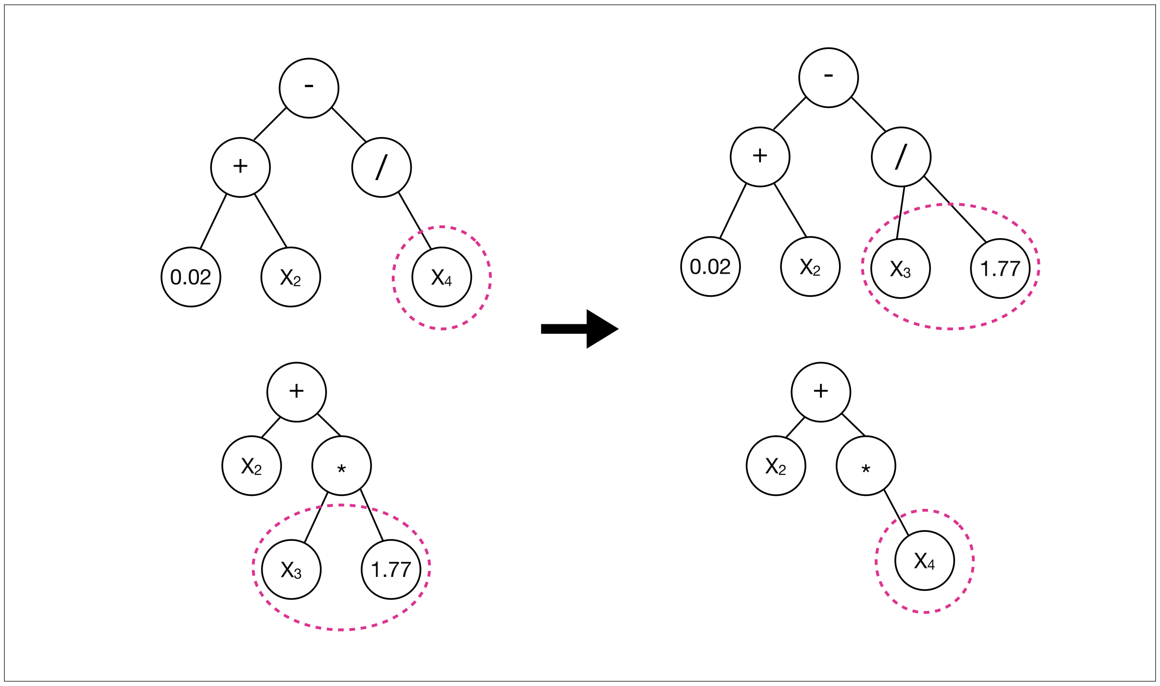


Fig. 3.20. The crossover operator between two trees in a GP population.

Algorithm 1: Genetic Programming pseudocode

```
for $i = 1$ to $NumOfGenerations$ (or until an acceptable solution is found) do
    if 1st generation then
        Generate the initial population with primitives (terminals, constants and
          math operations) usually with ramped $\frac{1}{2}$ and $\frac{1}{2}$ method;
    else
        With current population generate a new one using crossover and mutation
          operators;
    end
    Calculate fitness (minimize $mae$) of population members;
    Select $n$ members from population (tournament parameters) to participate in
      genetic operations according to specified probabilities;
    Create new individuals from genetic operations according to specified
      probabilities;
end
Result: Return best individual in last population
```

3.5 Performance Metrics

Everything that is to be improved it necessarily needs to me measured. To determine the efficacy of ML algorithms, a reference point must be established. Depending on the task at hand, the success of a method is dictated by how it performs against the expected results. For *classification* tasks a good judgment would be the percentage of successful samples that are correctly categorized. In the case of *regression* assignments the error with respect to the expected value is a reasonable assessment.

3.5.1 Classification metrics

There are several well accepted *classification* measures. Depending mostly on the dataset (i.e., this is balanced or not) and prediction task (e.g., mimicking a human behavior), either one or all of the next measurements can be employed:

3.5.1.1 The confusion matrix

The *Confusion Matrix* (CM) is a useful tool to asses in a deeper level the performance of a ML classifier. A CM is square matrix that plot the true labels against the predicted ones. The main diagonal of the matrix represents the correctly categorized samples. Everything off of this diagonal are either *false negative* of *false positive* instances. A CM can reveal a biased classifier and/or an unbalanced dataset. A "very accurate" classifier is not always as good as it seems, and a CM can reveal this [53]. See Fig. 3.21, where a CF is shown for the performance of a linear SVM classifier on the UCI wine dataset [83].

3.5.1.2 Accuracy

Accuracy can be viewed as a raw relation between successful and unsuccessfully classified instances. Nonetheless there is more than meets the eye, Accuracy can be broken down into *precision* and *recall* measures. First, *precision* can be acknowledged as the *true Positives* (TP) related to the number of true and false positives (TP, FP), whilst *recall* is the number of *true Positives* related to the combination of true positives plus false negatives (FN). Thus, *Accuracy* can be established as the fraction of all correctly categorized instances among all samples (see Eqs. 3.17, 3.18 and 3.19) [53].

$$Precision = \frac{TP}{(TP + FP)} \tag{3.17}$$

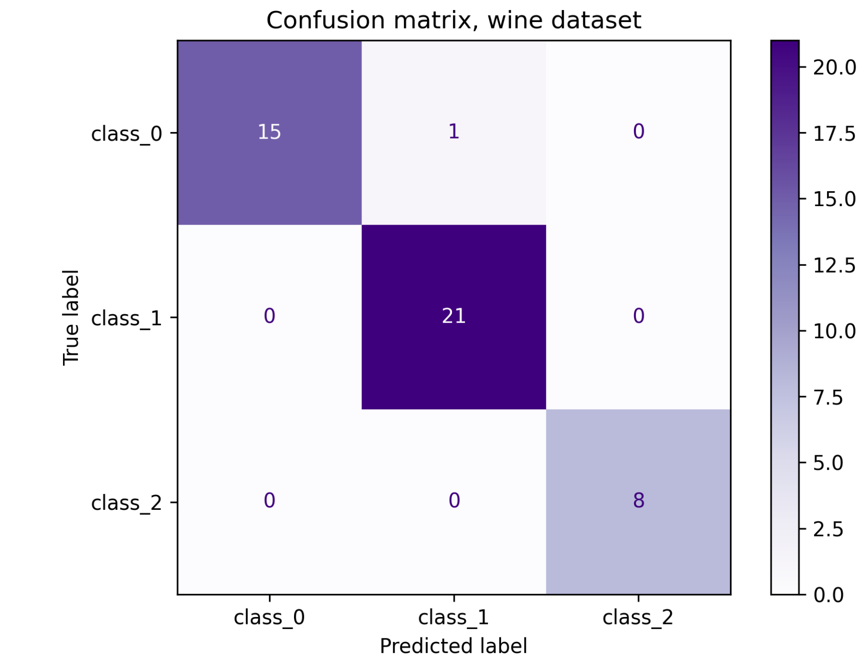


Fig. 3.21. A confusion matrix for the UCI wine dataset.

$$Recall = \frac{TP}{(TP + FN)} \tag{3.18}$$

$$Accuracy = \frac{(TN + TP)}{(TP + FN + FP + TN)} \tag{3.19}$$

Moreover, the *F1 score* is a stronger measure to asses classification problems. It takes into account equally the *precision* and *recall* measure to give a more balanced measure. It often delivers lower values than the *accuracy* by itself, but it is considered a more trusted appraisal. Mathematically the *F1 score* is actually the harmonic mean of the *precision* and *recall* values [53]. Eq. 3.20 shows how to compute the *F1 score*.

$$F1score = 2 \cdot \frac{(Precision \cdot Recall)}{(Precision + Recall)} \tag{3.20}$$

3.5.2 Regression metrics

To evaluate algorithms in *regression* problems we need to estimate the distance from the predicted value to the ground truth. We call this the *error*. There are several *error* measures that serve different purposes depending on the problem:

3.5.2.1 The error

Several error measures are available depending on the context they are intended to be used on. For example the *Mean Absolute Error* (MAE), also known as the *Manhattan* norm (ℓ_1) is computed using Eq. 3.21. In addition, the *Root Mean Square Error* (RMSE) or ℓ_2 norm is calculated with Eq. 3.22. To guide the election of the proper error measure we can consider that, when the norm index value (ℓ_i) tends to be larger, the more it focuses on greater values and disregards smaller ones [69, 53].

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \mid y_i - \hat{y}_i \mid \tag{3.21}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\Big(y - \hat{y}_i\Big)^2} \tag{3.22}$$

3.5.3 *k* fold cross validation

Validating a ML predictor algorithm requires several datasets. Often times the availability of various datasets is restricted to just a few or the one. To make up for the lack of data, a technique called *cross validation* is employed. Several variants of this method exist, being the *k fold cross validation* one of the most used and successful [53]. It consists of dividing the dataset in k partitions, where the k^{th} part is used as *test* and the rest as *train* data. The model is then fit and evaluated k times, then the *mean* or *median* of k *accuracy* or *error* values is reported. This technique communicates a more generalized depiction of the actual prediction model, since it reduces overfit and compensate for the shortage of datasets with k pseudo datasets

[53]. More often than not, the instances in a dataset might show some ordering, so before performing a k fold cross validation some type of shuffling must be performed on the data, in order to make sure that the *train* and *test* partitions will have enough samples of each class for the model to properly learn from. When the dataset is also unbalanced, a strategy known as *stratified shuffling* will randomly rearrange the samples, but making sure that the same proportion of each class appears in both the *train* and *test* partitions [69]. The most frequently used k values to partition the dataset are 3, 4, 5 and 10. Fig. 3.22 shows a 5 *fold cross validation* partition of a dataset.

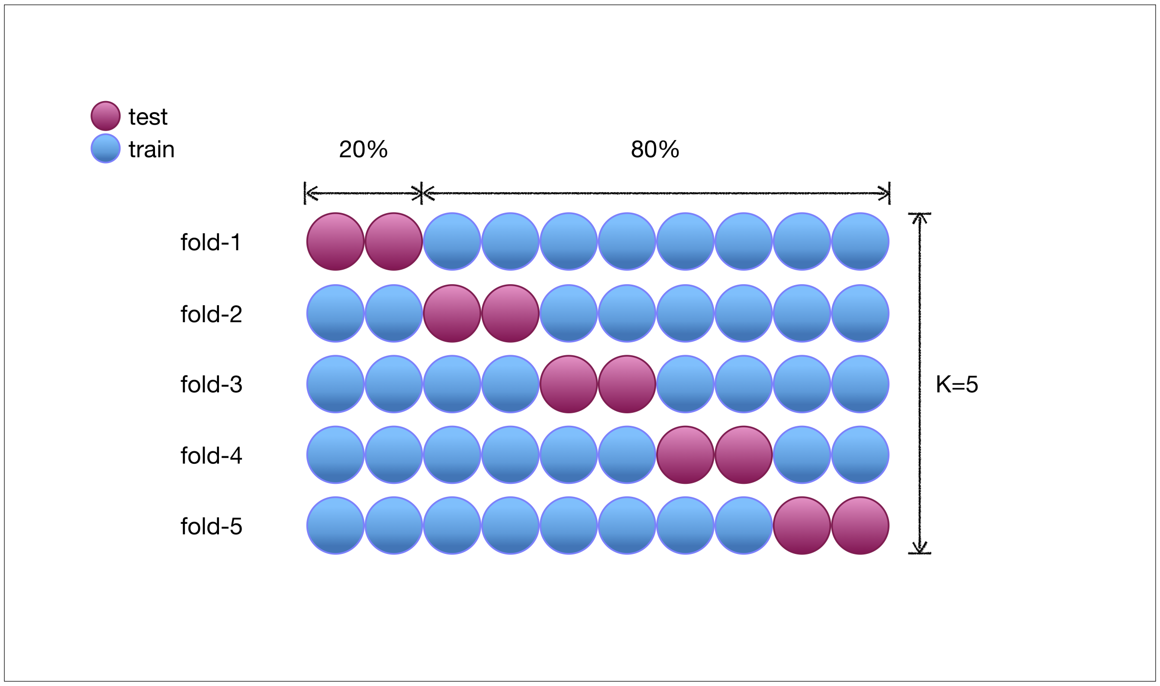


Fig. 3.22. A 5-fold cross validation example.

Figs. 3.6, 3.7, 3.8, 3.10, 3.11, 3.21 were inspired by graphics appearing in the book *"Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow"* by Géron, A. [69]. Figs. 3.18, 3.19, 3.20 were inspired by graphics in the book *"Machine Learning An Algorithmic Perspective"* by Marsland, Stephen [65]. The design of Fig 3.14 was taken from the *transformer* architecture presented in the paper *"Attention is all you need"* by Vaswani et al. in [68]

Chapter 4

A novel evolutionary-based term weighting-scheme for document embeddings

4.1 Motivation

Experts in areas such as *machine learning*, *computational linguistics* and even *psychology* work together frequently to solve the AP problem. In this study the purpose is to propose a novel and effective methodology that contributes to the State of the Art (SoA) in this field. As previously stated, the PAN scientific event organizes an annually shared task on *Author Profiling* since 2013. One main motivation to perform the study presented in this chapter, relied in the fact that the *test* partitions of the PAN datasets (2013-2018) were made publicly available circa late 2018. Up until that moment only the *train* partitions were available, which allowed studies to be carried out with a k-fold cross validation approach. In this regard, the comparison between the results achieved by the current work and the official results was made under equal circumstances.

In this study, the proposed method aims to produce Document Embeddings (DE) by means of evolving mathematical equations that integrate classical term frequency statistics. To accomplish this, a Genetic Programming (GP) strategy was employed to build competitive formulae to weight custom Word Embeddings (WEs), produced by cutting edge feature extraction techniques (e.g., word2vec, fastText, BERT). The rationale behind this novel methodology relies in producing DEs by averaging the WEs of the terms of a document with a tailor-made strategy. The aggregation techniques often employed in literature vary from being as straightforward as a simple mean of WEs, to the use of common frequency statistics (e.g., *tf-idf*) as weighting criteria. In this thesis it is hypothesized that task-specific weighting-schemes can be constructed, so more useful DEs can be produced for AP tasks. Moreover, the *relevant topic value* (*rtv*) statistic expressly devised for this study can capture latent information from the corpus. This statistic was able to encode thematic and word-usage information from texts, which enhances the weight of WEs associated with a specific target. Moreover, the *rtv* was selected by the GP algorithm as a useful variable more often than not, hence it can be suspected its outstanding usefulness.

4.2 Related Work

As mentioned early in Section 2.5, every year at the PAN event since 2013 new AP tasks and datasets are used, posing new obstacles. Take for instance the PAN 2015 competition, in this year the goal was to predict gender, age and 5 personality traits. Frequently the AP endeavour is posed as a *classification* task, nonetheless to predict personality (amount of trait present in each individual) most teams used a regression approach [16]. Furthermore, in the PAN 2016 event the AP task targeted the detection of gender and age in the English, Spanish and Dutch languages. The corpus was comprised of Twitter posts as the *train partition*, whilst blogs and other social media data for *test*, thus a cross-genre strategy aimed to produce models that could generalize better to unseen data [91]. In addition, in the 2017 shared AP task the objective was to classify gender and language variety for the English, Spanish, Arabic and Portuguese languages. For this competition, the dataset was composed with Twitter posts. The varieties used for each language were: *Australia*, *Canada*, *Great Britain*, *Ireland*, *New Zealand* and the *United States* for English. For the Arabic language, *Egypt*, *Gulf*, *Levantine* and *Maghrebi* subtypes were used. The Spanish part of the dataset was formed by *Argentina*, *Chile*, *Colombia*, *Mexico*, *Peru*, *Spain* and *Venezuela* strains. Moreover the regions used for the Portuguese language were *Brazil* and *Portugal* [92]. Finally in 2018 the AP task consisted only in the prediction of gender. The variant this year consisted in the choice to use images as well as textual corpora as features [15].

Frequently the top performer strategies in these shared tasks belong to the family of "traditional" Machine Learning (ML) algorithms. Take for instance the 2015 edition. For feature extraction some teams focused on the stylistic and thematic properties of the dataset, they chose Second Order Attributes (SOA) and Latent Semantic Analysis (LSA) to enhance the discerning skills of their algorithms [16, 93, 94]. Likewise, in 2016 the participants employed a mixture of stylistic features like n-grams and Part of Speech (POS) tagging [95]. Other competitors used second order representations, previously introduced in the original 2013 competition [96, 97]. It is also noteworthy the frequent use of Bag of Words (BoW), *tf-idf* weighted n-grams and as of lately WEs strategies [98, 99, 100]. Furthermore, in 2017 the best performer utilized a very straightforward method, consisting of a linear SVM with word unigrams and 3 & 5 character n-grams as features [46]. In addition, in 2018 the best performer using only textual features also implemented a very uncomplicated approach, consisting of a dimensionality reduction stage using LSA, word and character n-grams as features and a SVM as predictor [47].

4.3 Methodology

The objective in this chapter was to propose a novel technique for composing DEs and evaluate their suitability for different AP tasks. The DEs devised are to be processed by either a classifier or a regressor, producing a profile of the individuals by gender, age, and personality traits. Fig. 4.1 depicts the full proposal.

The methodology can be summarized in the following steps:

- Generate WEs for the datasets using the following feature extraction methods: word2vec algorithm (Skip-gram) [8], fastText [85] and BERT [64]. See item 1A in Fig. 4.1.

- Compute statistics for terms in each dataset vocabulary, (e.g., *tf, tf-idf, IG and rtv*). See item 1B in Fig. 4.1.

- Evolve mathematical equations through GP, using the values obtained in the last step as terminal variables, to compute the weights of the WEs. See item 2 in Fig. 4.1.

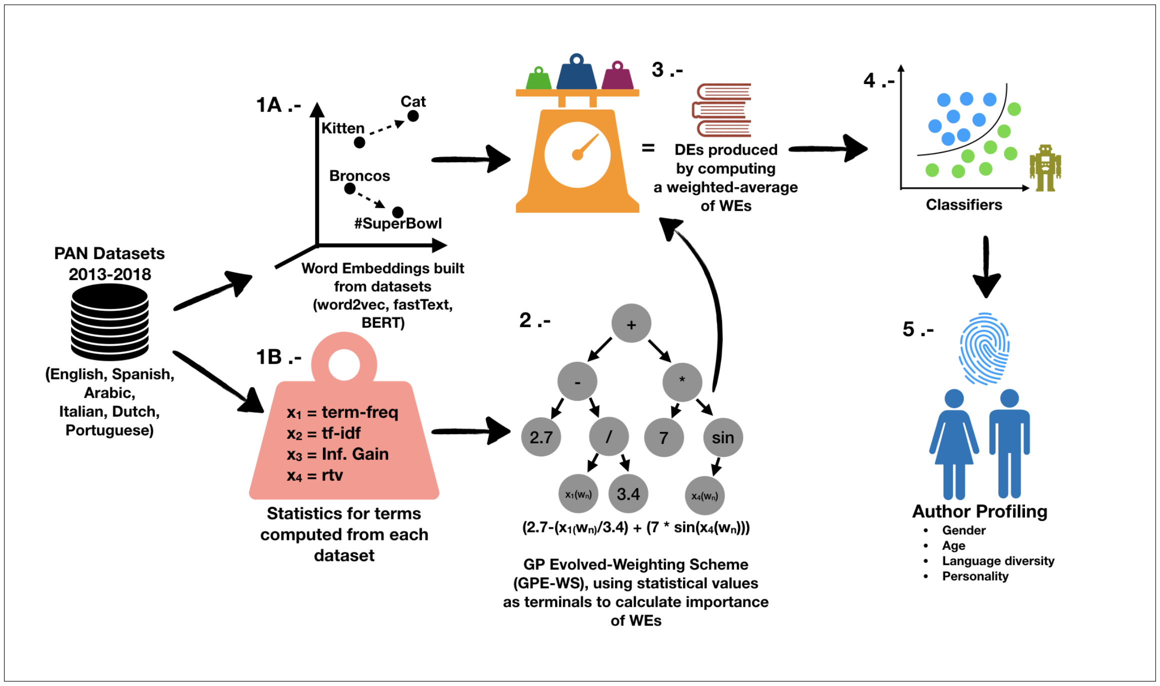

PAN Datasets 2013-2018
(English, Spanish, Arabic, Italian, Dutch, Portuguese)
1A .-
Kitten
Cat
Broncos
#SuperBowl
Word Embeddings built from datasets (word2vec, fastText, BERT)
1B .-
x_1 = term-freq
x_2 = tf-idf
x_3 = Inf. Gain
x_4 = rtv
Statistics for terms computed from each dataset
2 .-
+
-
*
2.7
/
7
sin
$x_1(w_n)$
3.4
$x_4(w_n)$
$(2.7-(x_1(w_n)/3.4) + (7 * \sin(x_4(w_n)))$
GP Evolved-Weighting Scheme (GPE-WS), using statistical values as terminals to calculate importance of WEs
3 .-
=
DEs produced by computing a weighted-average of WEs
4 .-
Classifiers
5 .-
Author Profiling
Gender
Age
Language diversity
Personality

Fig. 4.1. Architecture of the proposed approach: Document embeddings through a heuristic search

- Create DEs to represent user's documents by aggregating the WEs of their posts, using a weighted average with values computed in last step. See item 3 in Fig. 4.1.
- Predict gender, age, personality traits and language variety of users, using their DEs as features, with Machine Learning (ML) algorithms. See items 4 and 5 in Fig. 4.1.

All these steps are explained in more detail in the next sections.

4.3.1 Word Embeddings

Two approaches were employed to generate WEs. The first one is based on producing non-contextual distributed representations of words. For instance, the "apple" word is represented by only one vector, no matter its contextual meaning. Examples of this approach are Global Vectors (*GloVe*) [84], *fastText* [85] and *word2vec* (*Skip-gram*) [8]. The second approach produces contextual WEs, by using either a Bidirectional LSTM or a Transformer, which are trained using the context of words (left to right and vice versa). Since this is a DNN-based method, several strategies to combine the hidden layers must be examined, in order to select the one(s) with the most embedded knowledge. The BERT Language Model (LM) was employed for building contextual WEs, which are presumed to be superior than traditional word vectors when addressing polisemy. To extract the word vectors from the 12 hidden layers of BERT, the last four tiers were added, as suggested by the authors, since it is in those last layers were most of the contextual knowledge can be found [64].

A vocabulary of WEs was devised for each dataset. The only pre-processing done to the collections was tokenization and lower casing of the terms. No *stop words* were eliminated since they are believed to be useful in AP tasks. For non-contextual WEs three options were examined. GloVe vectors (offered as pre-trained data), although very effective in several documented problems, did not deliver optimal results for these particular tasks. Within Python *Gensim* library [101], word2vec and fastText algorithms were used for constructing non-contextual custom WEs (created from each dataset). Several configurations of parameters were explored, specifically the size (dimensions) and window (number of neighboring words). Table 4.1 shows the hyper-parameter settings examined. As the reader can see, small and large window values were considered (e.g., *5*, *40*). Furthermore, the best WE models were attained with window values of 30 and 40. Hence, it can be theorized that a large window is better suited for semantic, rather than lexical applications [8].

Table 4.1: Parameters examined for the word2vec (Skip-gram) and fastText algorithms

Word vector Alg.	Parameter	Value
word2vec (Skip-gram), fastText	‘size’	50, 100, 300 500
	‘window’	5, 10, 15, 30, 40

BERT was employed as a feature extraction method, using the *Transformers* library from Hugging Face [87]. The pre-trained BERT-Base (English and Multilingual models) allowed to encode 768-dimensional vectors, for the respective languages.

4.3.2 Statistical values to establish importance of terms

To establish the weight of a term within a document, a set of known and custom defined statistics were computed for each word in the vocabulary (i.e. term-frequency (*tf*), term-frequency inverse-document-frequency*tf-idf*, information gain*IG* and *rtv*). For this purpose, four different dictionaries of terms and their statistics were built using Eq. (4.1).

$$Dict \subseteq \{ \ (t, v) \ \mid \ f : T \rightarrow V \ \} \tag{4.1}$$

Where T is the set of terms that compose the vocabulary and V is a descending sorted list of values, associated with the relevance of each term in the set of documents of a given user. Table 4.2 display the four statistics used to construct the dictionaries of terms[1].

Table 4.2: Variables (terminals) used in the GP process to evolve weight equations

Dictionary	$f : T \rightarrow V$
Dict-x_1	$\text{tf}(w_n) = f(w_n)$
Dict-x_2	$\text{tf-idf}(w_n) = (1 + \log \text{tf}(w_n)) \cdot \log \frac{N}{\text{df}}$
Dict-x_3	$\text{IG}(w_n) = \sum_{n,y} P(w_n, y) \ln \frac{P(w_n,y)}{P(w_n)P(y)}$
Dict-x_4	$\text{rtv}(w_n) = T[i][w_n] + \log(\frac{N}{1+\text{ldf}}) + 1$

Next, the term statistics contained in the four dictionaries are described:

- Dict-x_1.- The frequency (occurrence) of each term within the documents (posts, tweets, etc,) of each user, where $f(w_n)$ is the raw count of a word (w_n) in a document. The more often a term appears, its value will be greater in the dictionary.

- Dict-x_2.- The *tf-idf* value, which increases according to the importance of the term across the dataset, where N is the total number of documents in the dataset and df is the number of documents containing the term. A term will have a greater value if it appears often in a few documents, whilst if a term appears often in many documents, its value will be offset, decreasing its amount.

[1]Dictionaries are noted as Dict-x_n, however when describing the value extracted from it, $x_n(w_m)$ or just x_n are used sometimes in favor of a clearer representation on a Table or Figure.

- Dict-x_3.- Establish the mutual information (dependence) between w_n (terms from a *tf-idf* sparse matrix) and y (the target class), where $P(w_n, y)$ is the probability of w_n given y, $P(w_n)$ is the probability of the vector of features and $P(y)$ the probability of the target. Terms that have a higher joint probability with its target, tend to score greater in the dictionary.

- Dict-x_4.- The *rtv* determines the importance of a term from the standpoint of content topic and personal usage of words. The matrix $T[i][w_n]$ represents the importance of a term within a topic, and it is produced by the Non-negative Matrix Factorization (NMF) method: $\mathrm{T} \approx \frac{F}{H}$, where F is a tf-idf sparse matrix of terms and H is a coefficient matrix. The NMF algorithm is used to extract *latent topics* from the corpora. Therefore, i represent the most likely topic a document belongs to, and w_n is the term in a document. Finally, $\log(\frac{N}{1+\mathrm{ldf}})+1$ is a personal smooth local *idf* offset, devised to numerically describe the personal usage of words, where N is the total number of documents of a given user, and ldf is the number of documents (posts) where the term appears. Terms that are more associated with their topic will have a greater value in the dictionary, offset by how users employ that word.

The *tf*, *tf-idf* statistics are often used in NLP tasks such as *textual similarity* or *theme segmentation*, whilst *IG* is mostly employed as a feature selection technique. In this work the dictionary of statistics Dict-x_4 was expressly devised for the AP task (*rtv*).

4.3.3 Term weighting equations evolved via Genetic Programming

Once the four dictionaries of importance values are assembled into a Statistical Dataset (SD), the GP algorithm evolves weighting-scheme equations. GP uses a symbolic regression approach to approximate a specific target. It constructs mathematical equations in the form of syntactic trees, employing *crossover* and *mutation* operators, to produce an initial population of random trees (equations), using values from the SD (previously built dictionaries) along with constants, as terminal nodes. Finally, a set of math operators are employed to mix the terminals into equations.

Fig. 4.2 depicts two examples of evolved equations, that were actually used in their corresponding datasets to compute the weight of terms.

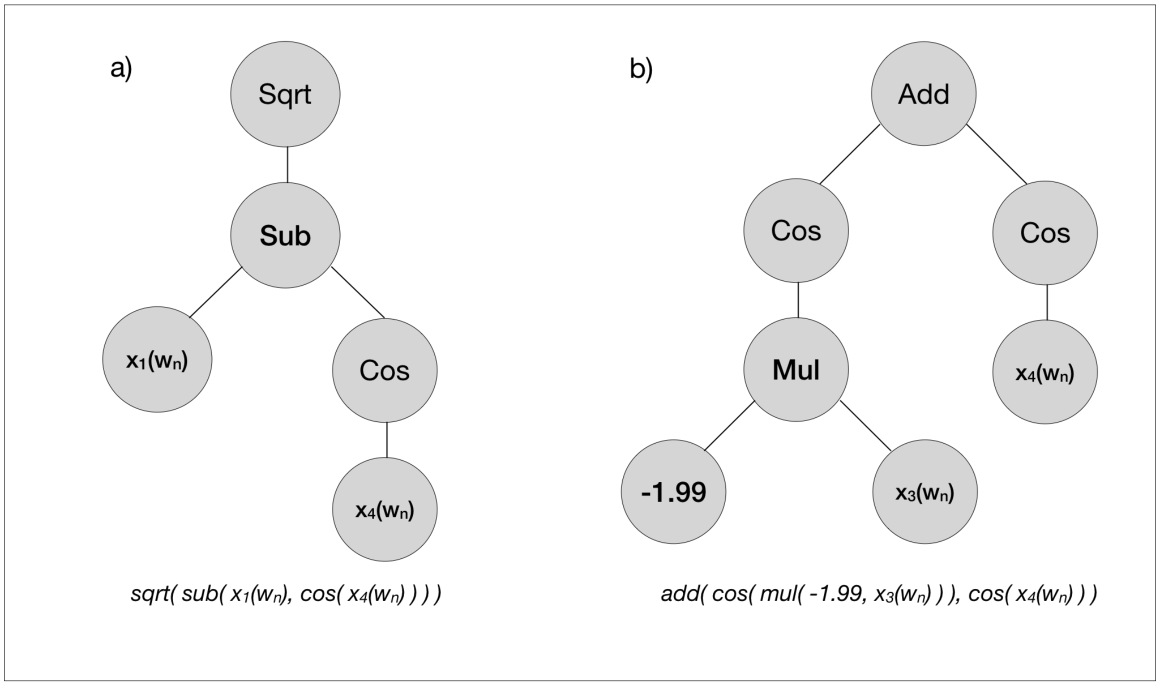

a)
Sqrt
Sub
x1(wn)
Cos
x4(wn)
sqrt(sub(x1(wn), cos(x4(wn))))
b)
Add
Cos
Cos
Mul
x4(wn)
-1.99
x3(wn)
add(cos(mul(-1.99, x3(wn))), cos(x4(wn)))

Fig. 4.2. GP generated equations. a) PAN-2018 Gender (Spa.) & b) PAN-2014 Gender Soc. Med. (Spa.)

The targets to approximate are the classes to be predicted in each dataset, previously converted into a tgt value where $tgt \in R = \{x \in R \mid x \geq 0\}$. In each passing generation, the most qualified (determined by a fitness function) individuals (equations) are passed to the next generation to participate in a new round of crossover and mutation operations, to produce a new population. New equations are generated by minimizing the fitness objective function *Mean Absolute Error* (MAE). These steps are repeated until an error threshold value, or a specific number of generations are reached. Finally, the best individuals in the last population are examined to select the one that delivers the best weighting scheme for WEs. Due to GP being a non-deterministic algorithm, a k-fold cross validation of the SD was conducted, exploring k the values of 3, 5, 7 and 10. Then, this strategy was executed at least 10 times in each dataset/task combination, to produce the best weighting schemes. Even though the algorithm was executed several times, a set of equations commonly made repeated appearances regardless of the fold or execution iteration. More often than not, the best equation belonged to the frequently appearing set of formulas. Moreover, there is very little difference in accuracy among all the final selected weighting schemes (formulas). Thus it can be theorized all of them are equally effective. For the GP implementation the *GPLearn* library [90] was employed. Table 4.3 shows the parameters used by GP. Algo. 2 depicts the whole GP stage.

Table 4.3: Hyperparameters of the GP Process

Parameter	Value
'Population size'	1000
'Generations'	100
'Function set'	'add', 'sub', 'mul', 'div', 'sqrt', 'log', 'sin', 'cos', 'tan', 'neg', 'max', 'min', 'abs'
'Metric'	MAE
'Tournament size'	20
'Constant range'	-10, 10
'Tree depths range'	2-6
'Init. method'	ramped half and half
'Crossover prob.'	0.9
'Mutation prob.'	0.1

The resulting evolved equations can be as short as two or three math operators and as many number of variables, but can also be very large, involving many operators and repeating many terminals. This phenomenon is known in GP as *bloating.* Although most of the attained equations are small, a few ones turned up rather large. Nonetheless, these broad equations did not affect the computational performance of the whole approach. The Eq. (4.2) is an example of a weighting-scheme for composing DEs, for predicting *gender* in the *Spanish* language for the PAN 2018 dataset (as seen in Fig. 4.2 item a)).

$$GPWeight(w_n) = sqrt(sub(Dict\text{-}x_1(w_n), cos(Dict\text{-}x_4(w_n)))) \tag{4.2}$$

Where $GPWeight(w_n)$ is the term weight used to aggregate the WE of the n-term into a DE. (w_n) is the n-term in a document. Dict-$x_1(w_n)$ is the *tf* value of the n-term, and Dict-$x_4(w_n)$ is the *rtv* score for the n-term.

Algorithm 2: A K-fold execution of the GP process, to produce the most competitive formulae.

```
for $i = 1$ to $K - folds$ do
    for $i = 1$ to $NumOfGenerations$ (or until an acceptable solution is found) do
        if 1st generation then
            Generate the initial population with primitives (terminals, constants
              and math operations) with ramped $\frac{1}{2}$ and $\frac{1}{2}$ method;
        else
            With current population generate a new one using crossover and
              mutation operators;
        end
        Calculate fitness (minimize $mae$) of population members;
        Select $n$ members from population (tournament parameters) to participate
          in genetic operations according to specified probabilities;
        Create individuals (equations) from genetic operations according to
          specified probabilities;
    end
    Return best individual in last population
end
Result: Return $K$ equations to find the best term weighting-scheme
```

4.3.4 Producing Document Embeddings via a weighted-average

As mentioned before, the methodology consists in producing DEs by aggregating single WEs into vectors of the same dimensionality. The DEs are computed using Eq. (4.3), whilst the two baselines used to compare against to are calculated with Eqs. (4.4) and (4.5).

$$\text{DEs-GPE-WS}(u_i) = \frac{\sum_{n=1}^{j} \big(GPWeight(w_n) * WE(w_n)\big)}{\sum_{n=1}^{j} GPWeight(w_n)} \tag{4.3}$$

$$\text{DEs-BL1}(u_i) = \frac{\sum_{n=1}^{j} \big(\text{tf-idf}(w_n) * WE(w_n)\big)}{\sum_{n=1}^{j} \text{tf-idf}(w_n)} \tag{4.4}$$

$$\text{DEs-BL2}(u_i) = \frac{\sum_{n=1}^{j} \big(WE(w_n)\big)}{n} \tag{4.5}$$

Where DEs-GPE-WS(u_i) is the DE, representing the posts of the i-user in the dataset, produced by a *Genetic Programming Evolved-Weighting Scheme (i.e., formulas, equations)* (GPE-WS), as it will be called for the rest of the chapter. DEs-

BL1(u_i) is the DE generated by using *tf-idf* as weighting-strategy. DEs-BL2(u_i) is a DE composed by a simple mean of WEs, j is the number of terms in documents of i-user, w_n is the n-term in the i-user's documents (posts). $WE(w_n)$ is the word embedding (vector) of the n-term in the documents. tf-idf(w_n) is the tf-idf value of the n-term, that can be retrieved from Dict-x_2. The reader must take special attention in $GPWeight(w_n)$ of Eq. (4.3), this value is obtained by using the GPE-WS for each $dataset \rightarrow target$ combination, as exemplified in Eq. (4.2).

4.4 Experimental Setup

Once the DEs are produced via a GPE-WS of WEs of documents, these are fed to four different ML algorithms to predict the *gender*, *age*, *personality traits* and *language variety* of users. *Classifier* and *Regressor* implementations of *Support Vector Machine* (SVM), *Random Forest* (RF), *Extra Trees*[2] (ET) and *K-Nearest Neighbors* (KNN) were used from the Python *Scikit-learn* library [82]. These ML algorithms were tested for evaluating the behavior of the proposed approach when used by several ML methods. Two rounds of experiments were conducted with these algorithms. First, a fixed set of hyper-parameters were used in all datasets/tasks to test the generalization of the approach. The results attained were competitive. Then, in a second round of experiments a hyper-parameter grid search for each collection in the train set was conducted, which obtained even better results. In section 4.5, the outcome achieved with the hyper-parameter tuned models is presented. The results from the first round of experiments are only included in Fig. 4.11, to show how both sets of experiments are competitive, thus it can be presumed the methodology is robust.

4.4.1 Datasets

The main purpose of this study was to devise and evaluate a representation of documents that could be practical in the AP problem. For this work several datasets crafted expressly for AP tasks were utilized. These collections were designed by the organizers of the PAN event at CLEF for every shared task since 2013. Most of the information of users are collected from social media outlets, blogs, Twitter and hotel

[2]Extremely Randomized Trees. Split nodes using a *random split* rather than a *best split* strategy

reviews. The aim is to predict the *gender*, *age*, *personality traits* and even *language variety* of each user in a test partition. All of these classes were approached as a classification problem, except personality traits. De to its nature this problem was posed as a regression task. Table 4.4 illustrates a summary of these datasets. For a more comprehensive explanation about these tasks and datasets refer to [32, 102, 16, 91, 92, 15].

These collections have been used in several studies in the AP problem, where a k-fold cross validation approach was used most of the times, because only the *train* partitions were available. Since the *test* partitions of these datasets were made public just recently, a competition strategy (separate *train* and *test* sets) could be done to compare the performance of the proposed architecture against the official results. Figs. 4.3 through 4.8 show the length in *tokens* (words), of *train* and *test* partitions in the 2013-2018 PAN datasets. Next, a brief description is given for each collection.

The main source of information for the 2013 dataset comes from outlets of blog posts such as Netlog, which were labeled with information about the author like gender and age. As can be seen in Fig. 4.3, most of the *train* and *test* samples are very short (composed of a few tokens), whilst a minority of them are large (more than 10,000 tokens).

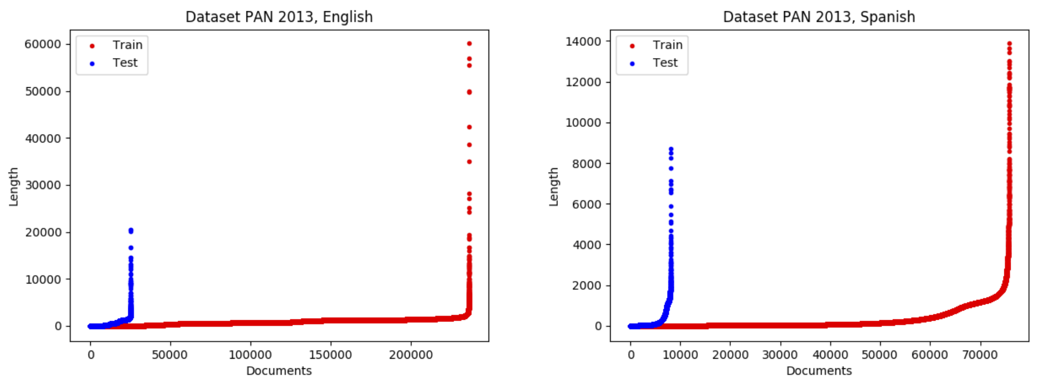


Fig. 4.3. *train* and *test* samples for the 2013 dataset.

In the 2014 AP task, the dataset was divided into four different sub-datasets (*social media*, *blogs*, *Twitter*, and *hotel reviews*). The purpose behind this idea was to observe the behavior of models across different types of information sources. For the *social media* option the data came from a subset of the previous year dataset (PAN 2013).

Table 4.4: Amount of samples in each dataset. Also, the average length of train and test documents is included. The length of a document is measured in number of tokens considered to build the DEs. The standard deviation is included for reference.

		Documents			***Train length stats***		***Test length stats***	
Dataset Lang.	***Classes***	***train***	***test***	***total***	***Avg.***	σ	***Avg.***	σ
2018 EN[a]	G[b]=2	3000	1900	4900	1597.24	287.60	1595.17	280.83
SP[c]		3000	2200	5200	1532.18	329.87	1540.45	334.10
AR[d]		1500	1000	2500	1399.93	454.06	1399.15	455.77
2017 EN	G=2, V[e]=6	3600	2400	6000	1592.89	293.15	1589.65	289.63
SP	G=2, V=7	4200	2800	7000	1517.35	336.46	1528.60	342.18
AR	G=2, V=4	2400	1600	4000	1439.25	440.95	1432.35	450.95
PT[f]	G=2, V=2	1200	800	2000	1217.70	321.44	1219.01	302.62
2016 EN	G=2, A[g]=5	436	78	514	9139.99	6571.51	4295.04	5935.62
SP	G=2, A=5	250	56	306	12558.47	6630.72	3883.75	8725.68
DU[h]	G=2	384	500	884	2651.71	1217.71	153.97	74.31
2015 EN	G=2, A=4, P[i]=5	152	142	294	1324.14	411.28	1312.45	429.15
SP	G=2, A=4, P=5	100	88	188	1613.36	311.27	1610.26	349.50
DU	G=2, P=5	34	32	66	1428.12	361.32	1257.28	339.21
IT[j]	G=2, P=5	38	36	74	1481.16	278.00	1420.56	277.48
2014 SM[k] EN	G=2, A=5	7746	3376	11122	10560.81	5305.88	13561.03	5560.58
SP	G=2, A=5	1272	566	1838	6392.58	6488.43	9708.52	8652.81
2014 B[l] EN	G=2, A=5	147	78	225	4437.62	7131.94	4295.04	5935.62
SP	G=2, A=5	88	56	144	8238.98	11982.67	3883.75	8725.68
2014 T[m] EN	G=2, A=5	306	154	460	9028.76	6960.88	9253.54	6271.77
SP	G=2, A=5	178	90	268	13124.87	7250.98	12025.30	6347.86
2014 R[n] EN	G=2, A=5	4160	1642	5802	271.04	350.97	249.65	303.49
2013 EN	G=2, A=3	236600	25439	262039	843.82	678.59	584.92	739.06
SP	G=2, A=3	75900	8160	84060	300.01	614.81	265.45	533.65

[a]English
[b]Gender
[c]Spanish
[d]Arabic
[e]Variety
[f]Portuguese
[g]Age
[h]Dutch
[i]Personality
[j]Italian
[k]Social Media
[l]Blogs
[m]Twitter
[n]Reviews

In the case of the *blogs* and *Twitter* subsets the information was manually selected and annotated by the organizers. Finally, the *hotel reviews* subset was derived from the Tripadvisor website. As the reader can observe in Fig. 4.4, the *social media* follows a pattern similar to the 2013 dataset, since it was taken from it. Also the *hotel reviews* dataset presents a similar design, whilst the *blogs* and *Twitter* subsets are more balanced with respect to the number of samples.

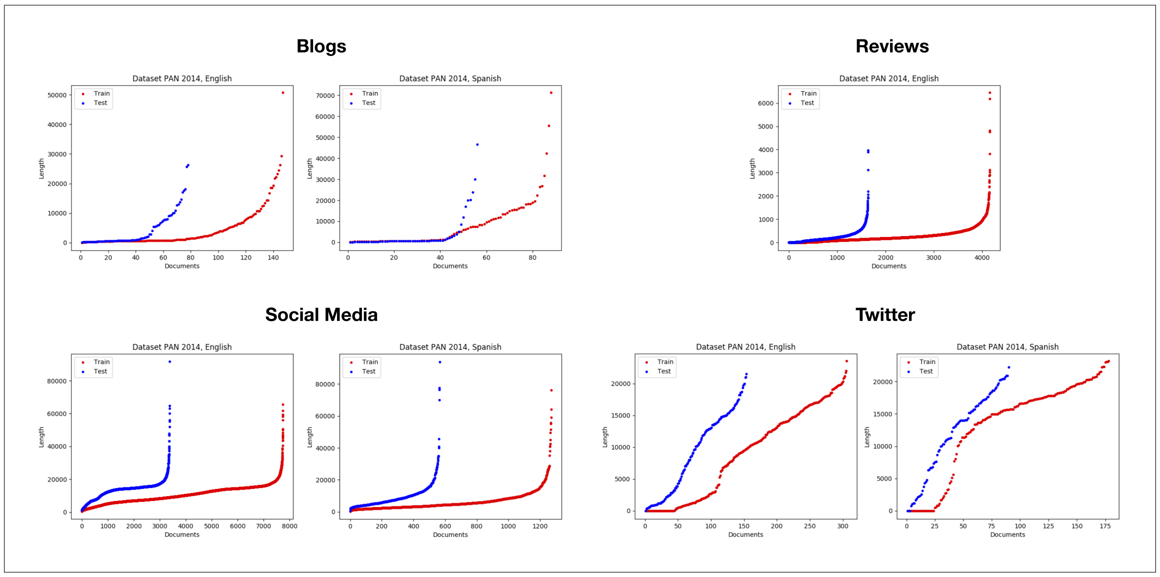


Fig. 4.4. *train* and *test* samples for the 2014 dataset.

The 2015 PAN dataset was formed by user posts from the Twitter platform. The samples were manually annotated with age and personality scores. The personality was self assessed by the users using an online test. Fig. 4.5 shows the outline of this dataset. Even though the number of samples is considerably smaller than previous years, the length of the posts is more balanced.

The 2016 AP task intended a cross-genre evaluation, that is, the *train* and *test* partitions came from different sources. For the *train* portion the information was obtained from Twitter posts, while for the *test* partition the samples were taken from blogs. Fig. 4.6 shows moderately balanced datasets for the English and Spanish languages, whilst for the Dutch idiom, we can observe that the *test* segment is considerably larger than the *train* set, which is atypical.

The 2017 and 2018 datasets were also comprised form Twitter posts. In the 2017 edition the task consisted of the prediction of gender and language variety (different regional variants of English, Spanish, Arabic and Portuguese). In 2018 only the gender was the forecasted target. Figs. 4.7 and 4.8 shows the datasets for these years are more balanced with respect to the length of the posts.

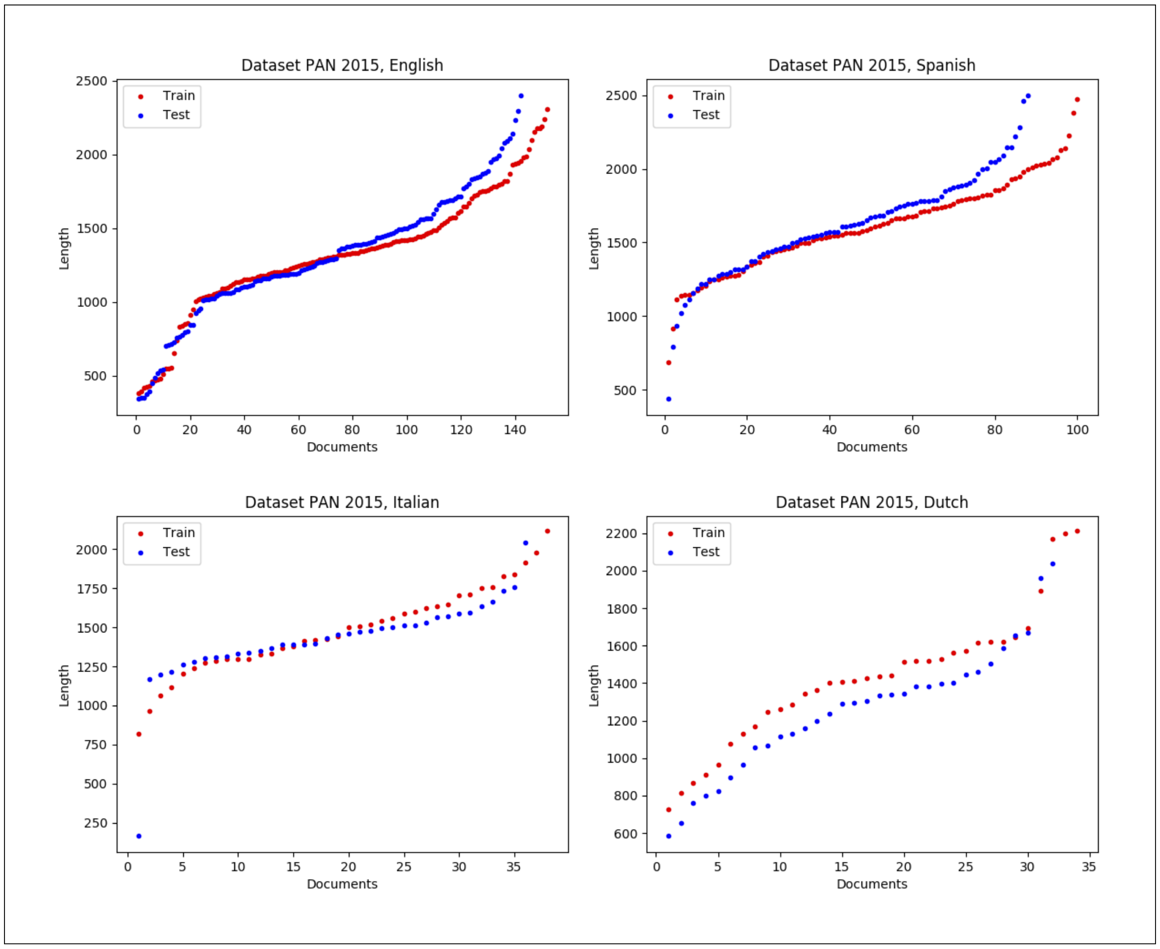


Fig. 4.5. *train* and *test* samples for the 2015 dataset.

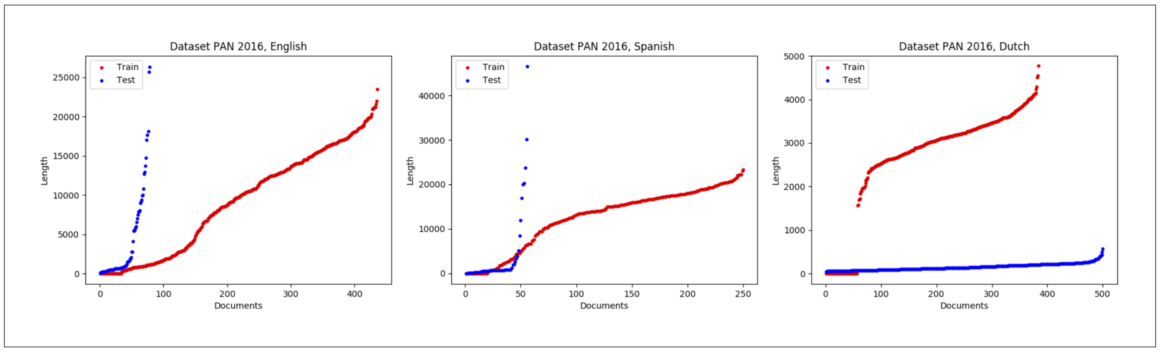


Fig. 4.6. *train* and *test* samples for the 2016 dataset.

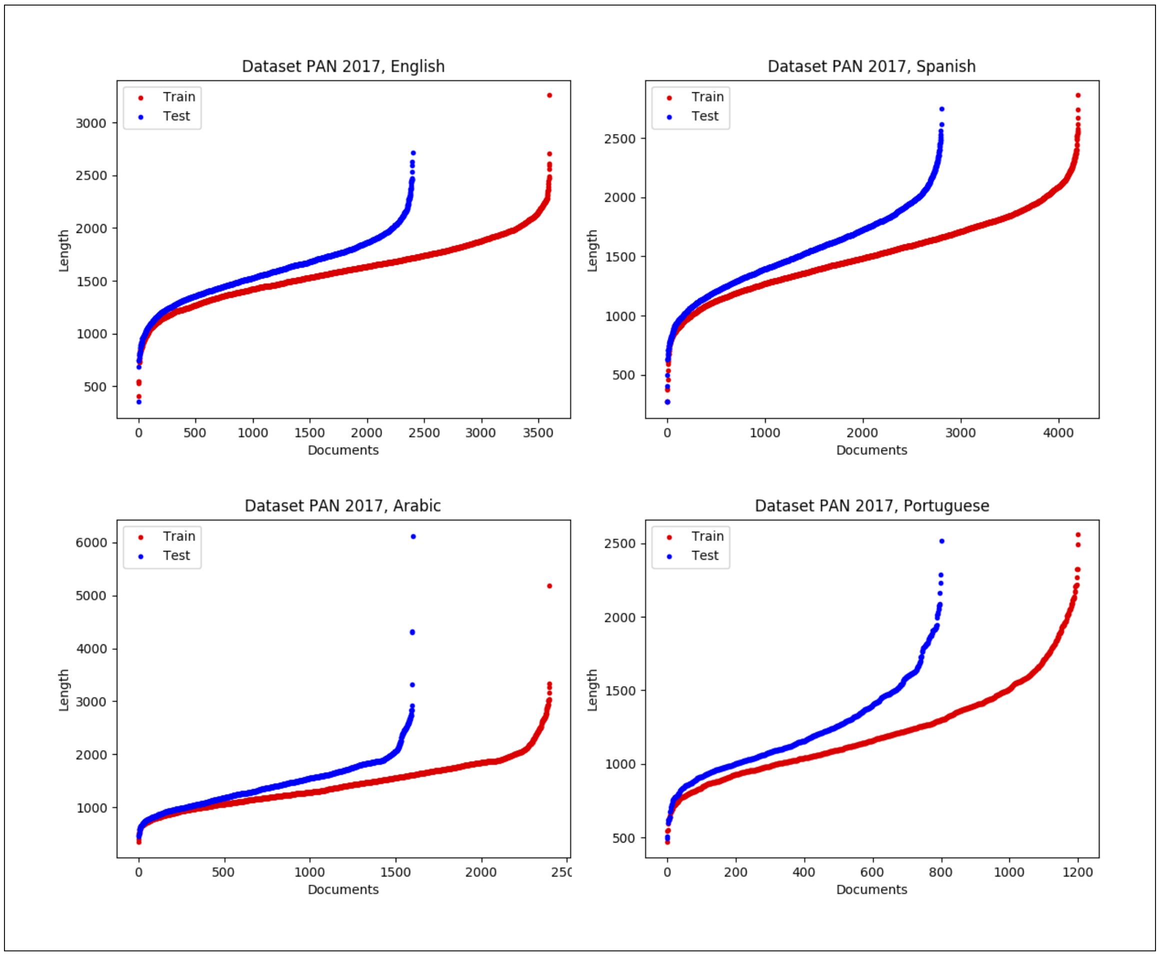


Fig. 4.7. *train* and *test* samples for the 2017 dataset.

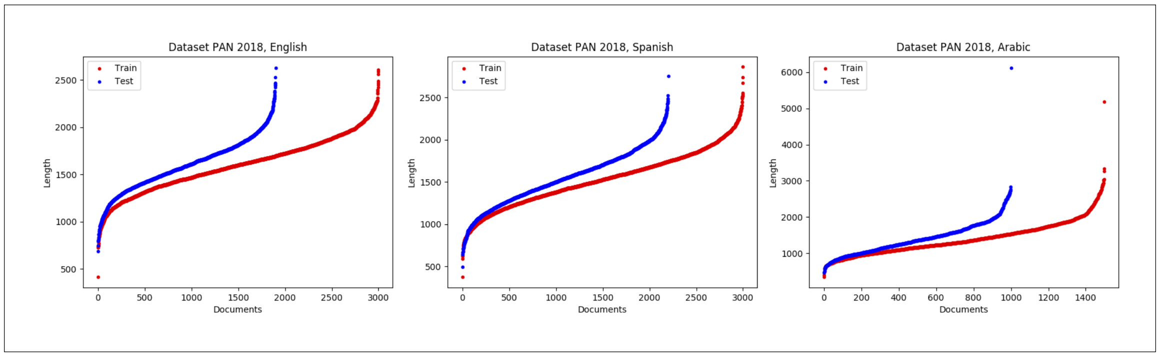


Fig. 4.8. *train* and *test* samples for the 2018 dataset.

4.5 Results

This section is organized in two main sets of experiments. In the first experiment the performance of the GPE-WS approach was evaluated with regard to two well known baseline strategies and the proposed statistic value: *a)* a *tf-idf* weighted-average, *b)* a WE regular average and *c)* a *rtv* weighted-average. The aim of this experiment is to contrast the performance of this proposal against other common methodologies to generate DEs. For the second experiment, the proposed methodology was compared against the winning teams from years 2013-2018 in the AP shared tasks at PAN.

4.5.1 GPE-WS vs Baselines

Table 4.5 shows gender prediction, as measured by Accuracy and F-1 score, through all PAN competitions for the GPE-WS approach and the baselines. The reader can observe that the GPE-WS consistently outperforms the baselines, even in some years with an important difference in performance (take for instance year 2015). Only in 2013 the *baseline 1* (*tf-idf* average) surpassed GPE-WS, and barely by a narrow margin. This can be partially explained due to the short nature of the posts in this dataset, which could cause limited topic understanding weighting schemes.

Table 4.5: Gender prediction accuracy by year at PAN (the shown results are the average of all languages in each dataset).

	GPE-WS		Baseline 1		Baseline 2		rtv	
Dataset	Acc.	F-1.	Acc.	F-1.	Acc.	F-1.	Acc.	F-1.
2013	0.6209	0.6181	**0.6213**	**0.6191**	0.6187	0.6161	0.6095	0.6074
2014-SM	**0.6084**	**0.6076**	0.5844	0.5836	0.5360	0.4507	0.5875	0.5836
2014-BL	**0.7214**	**0.7146**	0.6369	0.6247	0.5000	0.3333	0.6229	0.6002
2014-TW	**0.7804**	**0.7801**	0.6433	0.6426	0.6494	0.5655	0.6674	0.6673
2014-RW	**0.6827**	**0.6826**	0.6797	0.6796	0.6815	0.6814	0.6711	0.6711
2015	**0.9100**	**0.9099**	0.7957	0.7870	0.8368	0.8343	0.8115	0.8021
2016	**0.6882**	**0.6853**	0.6264	0.6203	0.6204	0.5968	0.6477	0.6388
2017	**0.8087**	**0.8086**	0.7774	0.7771	0.7906	0.7905	0.7895	0.7894
2018	**0.8047**	**0.8046**	0.7835	0.7834	0.7925	0.7924	0.7911	0.7910

Table 4.6 presents the comparison of the same strategies for predicting age in seven different datasets. Still, each year, GPE-WS offers very competitive results, then suggesting it can be a strong approach to generate DEs for two common AP variable predictions.

Table 4.6: Age prediction accuracy by year at PAN (the shown results are the average of all languages in each dataset).

	GPE-WS		Baseline 1		Baseline 2		rtv	
Dataset	Acc.	F-1.	Acc.	F-1.	Acc.	F-1.	Acc.	F-1.
2013	**0.6662**	**0.4515**	0.6629	0.4475	0.6627	0.4490	0.6546	0.4416
2014-SM	**0.4358**	**0.3291**	0.3890	0.2786	0.3104	0.1799	0.3862	0.2806
2014-BL	**0.5549**	**0.2630**	0.3265	0.1679	0.3860	0.1105	0.3315	0.1638
2014-TW	**0.5556**	**0.3212**	0.5227	0.3010	0.5088	0.2102	0.4953	0.2933
2014-RW	**0.3167**	**0.1828**	0.3143	**0.1828**	0.3149	0.1820	0.3124	0.1799
2015	**0.8343**	**0.7613**	0.7686	0.6660	0.6717	0.5863	0.7466	0.6771
2016	**0.5614**	**0.3235**	0.4208	0.1855	0.4780	0.2460	0.4425	0.1987

Besides *gender* and *age*, in 2015 also personality traits were predicted. The targets for personality are *Openness (to experience)*, *Conscientiousness*, *Extroversion*, *Agreeableness* and *Neuroticism/Stableness*, as proposed by the *Five Factor Model* (FFM) of personality [103]. As mentioned before, this problem was posed as a regression task. The values to estimate vary between certain ranges to predict the amount of such trait present in a specimen. In this case the evaluation measure used was the *Root Mean Square Error* (RMSE), as shown in Eq. (4.6). Fig. 4.9 shows the comparison between the GPE-WS against baselines in terms of RMSE. As the lector can observe, GPE-WS also achieves top performance when predicting all single personality traits against common baselines. Also note that *rtv* by itself obtains comparable results with regard to the other two baselines.

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{n}\Big(predicted_i - target_i\Big)^2}{n}} \tag{4.6}$$

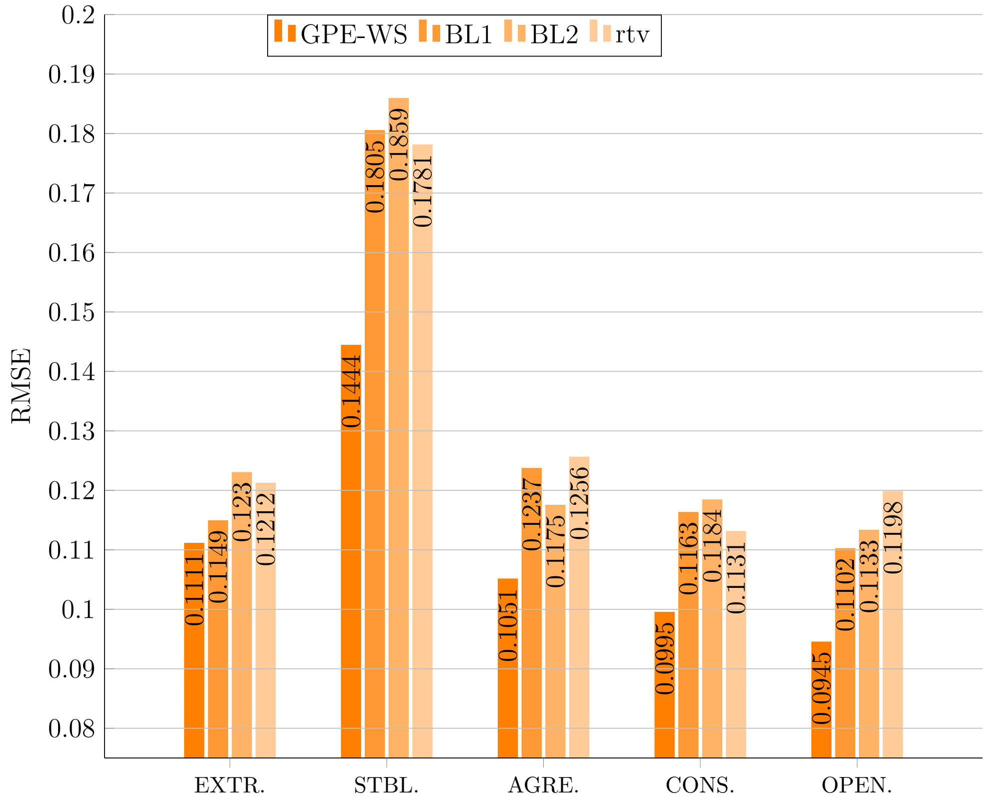


Fig. 4.9. Personality prediction (RMSE) at PAN 2015 (the shown results are the average of all languages in each personality trait).
(***lower is better***)

For the 2017 shared task, another AP variable was involved, language diversity. Here, the goal was to predict regional variants of the English, Spanish, Arabic and Portuguese languages. Fig. 4.10 shows the comparison of the GPE-WS against same baselines for this specific task. We can still observe that even for granular sub-tasks the results reported for the proposed approach are very consistent.

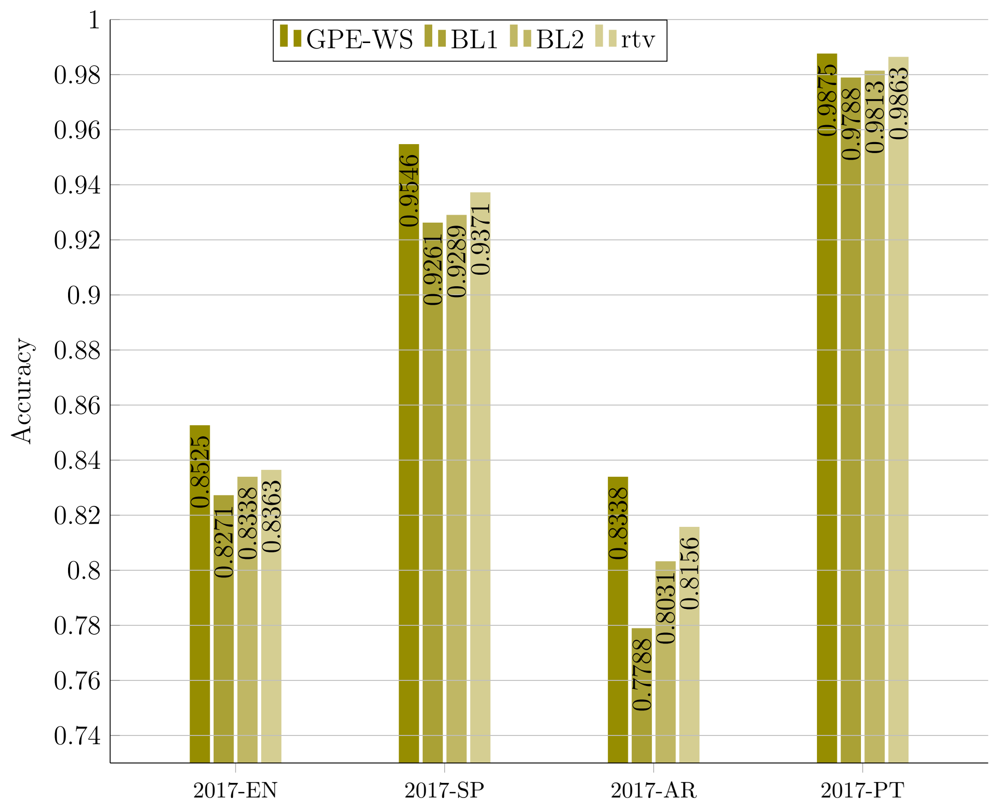


Fig. 4.10. Language diversity prediction (Accuracy) at PAN 2017. (***higher is better***)

4.5.2 GPE-WS vs PAN 2013-2018 Competitors

Table 4.7 presents a direct comparison between the accuracy results obtained by the GP approach against the best performer (winner) at each year competition at PAN. It is important to notice the variety of dataset-specific feature engineering strategies among the top participants, yet the intended approach scores in the top-quartile along with the first places. Moreover, in Fig. 4.11 we can appreciate the entire perspective. As can be seen, the strategy submitted in this study shows a consistent performance among the top achievers in every event.

Table 4.7: Accuracy of GPE-WS vs. best competitor at PAN shared tasks (2013-2018). The shown results are the average of all languages for each dataset/task

Dataset	GPE-WS	Best @PAN
2013-G	**0.6213**	0.5995
2013-A	**0.6662**	0.6565
2014-SM-G	**0.6084**	0.5834
2014-SM-A	**0.4358**	0.4038
2014-BL-G	**0.7214**	0.6344
2014-BL-A	**0.5549**	0.4398
2014-TW-G	**0.7804**	0.6604
2014-TW-A	**0.5556**	0.5134
2014-RW-G	**0.6827**	0.6809
2014-RW-A	0.3167	**0.3337**
2015-G	**0.9100**	0.8712
2015-A	**0.8343**	0.8168
2015-Extro[a]	0.1111	**0.1018**
2015-Stable[a]	**0.1444**	0.1581
2015-Agree.[a]	0.1051	**0.0736**
2015-Consc.[a]	**0.0995**	0.1151
2015-Open[a]	**0.0945**	0.0946
2016-G	**0.6882**	0.6184
2016-A	**0.5614**	0.5538
2017-G	0.8087	**0.8253**
2017-V	0.9071	**0.9184**
2018-G	0.8047	**0.8170**

[a] Lower is better.

As mentioned before, four different algorithms were used to predict characteristics from authors, using their DEs as features. Fig. 4.12 describes the achievement of each classifier predicting *gender* and *age* (averaged in all languages). The reader can observe that *SVM*, *RF* and *ET* attained the best results, 37.5%, 37.5% and 25% of the times, respectively. As can be noted, the difference in performance between them is little; therefore it can be presumed certain robustness of the methodology.

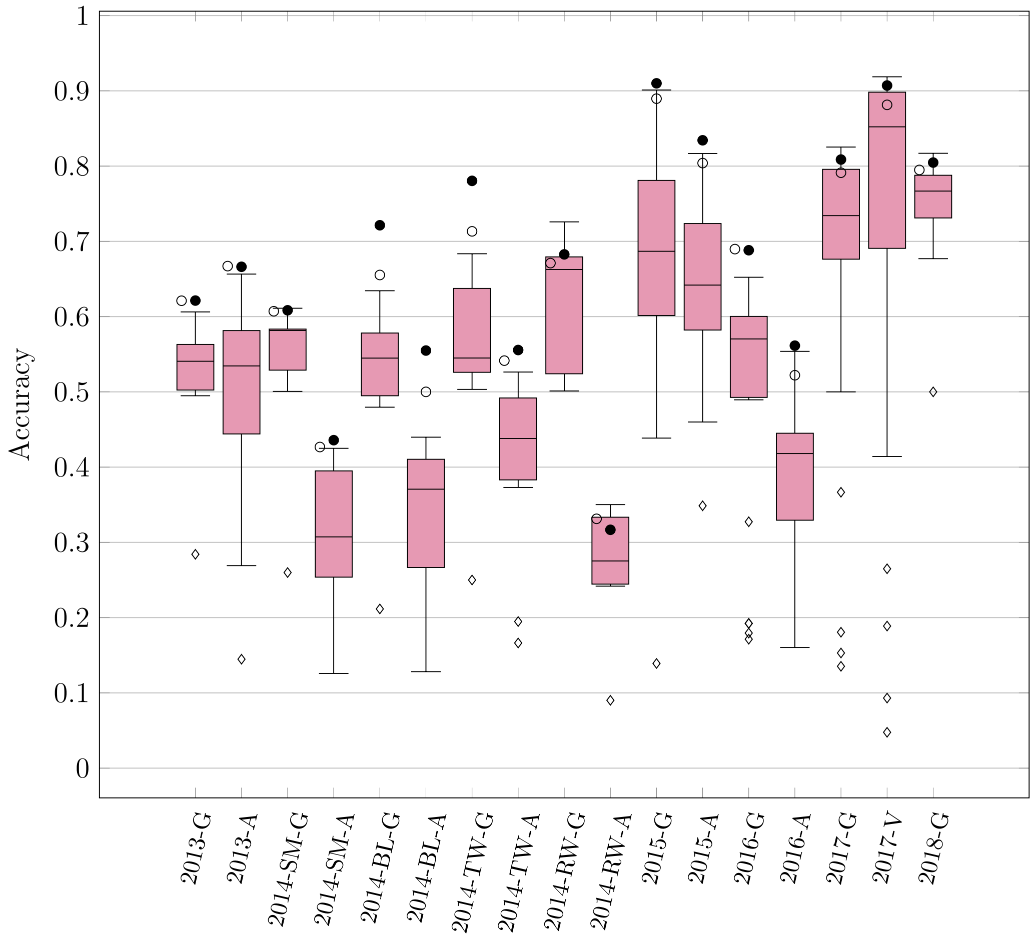


Fig. 4.11. Boxplots depicting the performance of GPE-WS vs all competitors at PAN (2013-2018). A black filled circle represents GPE-WS results with hyper-parameter search, and a blank circle depicts GPE-WS with a fixed hyper-parameter set for WE models and classifiers. The shown results are the average of all languages for each dataset/task. Diamonds portrait the outliers.

4.5.3 *rtv*: A New Term Weight for Author Profiling

The *rtv* statistic devised for this work, was an important contributor in the GPE-WS, due to its capability to enhance importance of *terms* across the built DEs. Take for instance Table 4.8, which shows the frequency of the variables (terminals) more often employed by GP to construct the best equations. Similarly, Table 4.9 shows an even more revealing picture, the feature most likely to appear alone in a single equation is $x_4(w_n)$, which happens to be the *rtv* that was specifically developed in this study for AP tasks. In addition, the combination of features more often appearing are $\{x_2 - x_3\}$, $\{x_2 - x_3 - x_4\}$ and $\{x_2 - x_4\}$, thus suggesting that a blend of these variables also

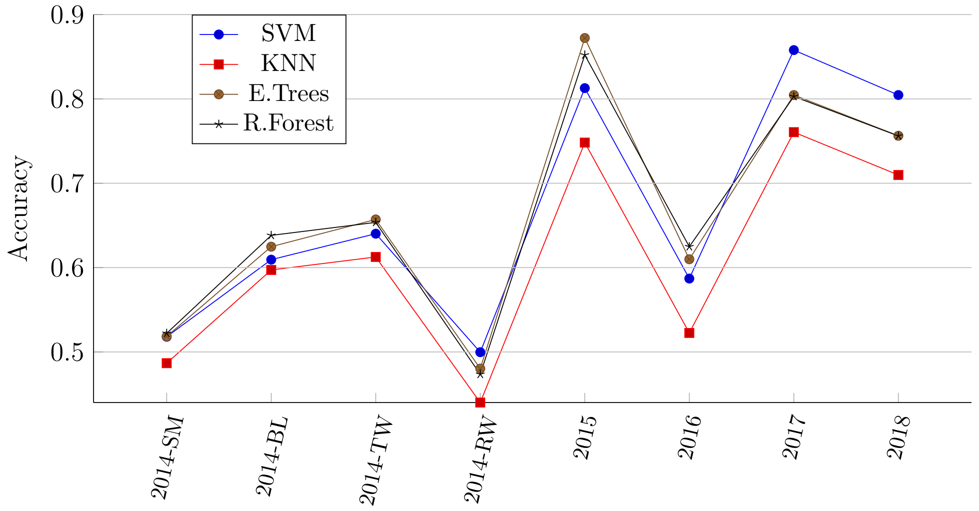


Fig. 4.12. Classifier's accuracy performance, using GPE-WS DEs in Gender & Age (average in all languages) prediction by year at PAN. (2017 is the average of Gender & and Language variety predictions, and 2018 gender only).

delivers effective results. Thematic information is crucial for AP tasks [104]. This might be one possible explanation for the success of the *rtv*, since this statistic was devised to enhance the weight of terms more associated with the prevalent theme in a user's document. Also, this metric was also rewarded in proportion with rarity of words, established by an *inverse document frequency* (idf) value computed for each user's document, to measure the usage of words by each person, thus providing an additional discerning boost to the *rtv*.

Table 4.8: Frequency of the terminal set in the GPE-WS

Terminal	Appearances	%
$x_1(w_n)$	1	0.47
$x_2(w_n)$	108	50.23
$x_3(w_n)$	47	21.86
$x_4(w_n)$	59	27.44

Table 4.9: Frequency of terminal combinations in the GPE-WS

Terminal combination	Appearances	%
$x_1 - x_2$	1	1.67
x_2	3	5.00
$x_2 - x_3$	11	18.33
$x_2 - x_3 - x_4$	6	10.00
$x_2 - x_4$	5	8.33
x_3	2	3.33
$x_3 - x_4$	1	1.67
x_4	31	51.67

4.5.4 Performance and size of datasets

Performance of ML algorithms and size of datasets are intimately related. While the amount of samples has been proved to be very important, the size of each instance must be examined too, specially in complex NLP tasks such as AP. In the present study, it was explored whether there was a correlation between performance and the length of the documents (measured in the number of tokens used to produce DEs). The analysis indicates that no clear correlation exists between accuracy and average size of documents in *train* and *test* sets. This finding is compatible with previous works, where the size of textual samples did not affect considerably the overall efficiency, as long as the number of samples were numerous [16, 38]. Fig. 4.13 shows no discernible correlation exists between *document size* and *accuracy*, in either *train* and *test* partitions, for *gender* and *age* predictions.

4.5.5 Contextual vs Non-Contextual Word Embeddings

As previously stated, the feature extraction phase was addressed with two types of WE techniques. For the non-contextual WEs, *word2Vec* and *fastText* algorithms were examined. From the values investigated for both approaches (see Table 4.1), dimension sizes of 300 and 500, as well as window sizes of 30 and 40, delivered the best results. Although there is no clear pattern to determine in which type of datasets each performs better, it is suspected *fastText* models might have succeeded in datasets with substantial out-of-vocabulary words, since this algorithm is trained with character level information. In both cases, they performed the best most of the times

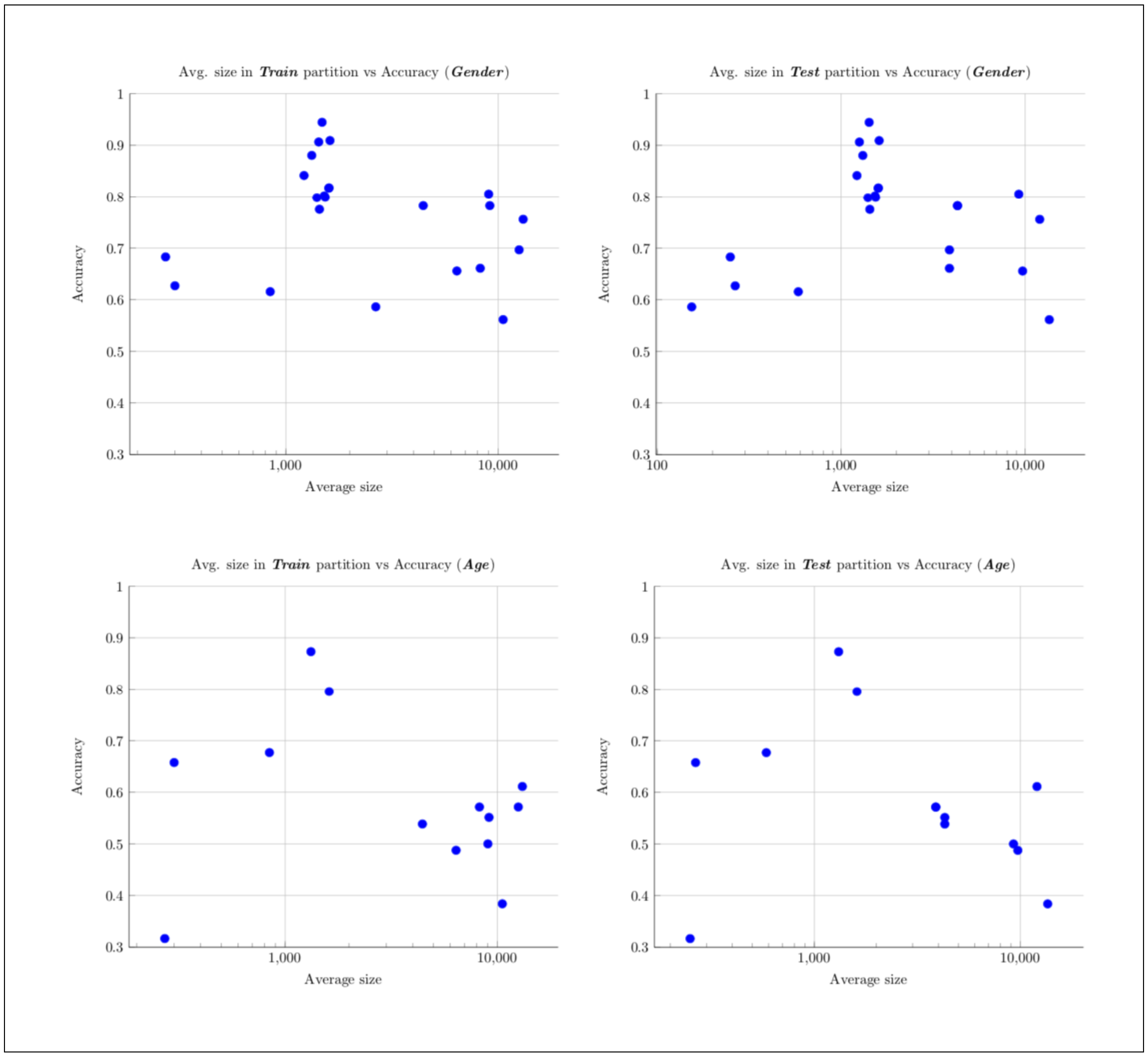


Fig. 4.13. Graphics showing ***no*** evident correlation between *Accuracy* and *Document Size*, in *train* and *test* partitions, for *gender* and *age*.

in non-English datasets. Furthermore, a value greater than 500 normally increased the computational time required for training, with no useful gain in accuracy, whilst a dimension lesser than 300 saw a clear decline in performance. Likewise it can be hypothesized a large window (as opposed to small ones like 5) suits better for semantic tasks such as AP.

In the case of contextual WEs, BERT was used as a feature extraction method to compose word vectors. The theoretical benefits of such approach have already been discussed. In English datasets BERT achieved the best results more times than any other model. This can partially be explained since the BERT pre-trained model for this language was prepared with the BooksCorpus [105] and the whole English Wikipedia, while the multi-language pre-trained BERT was processed with the not-as-vast Wikipedia corresponding languages. Even though BERT was not trained on a *social media* corpus, it is worth noting that in some cases, when BERT performed better than *word2vec* and *fastText* models for English datasets/tasks, it did so with a clearer margin. Thus, it can be presumed that an English or Multilingual BERT model, trained over larger datasets from more diverse contexts (e.g., social media, medical corpus, product reviews), could potentially outperform contextual WEs in AP related problems. See Table 4.10 for a complete reference of each model used to produce WEs in the dataset/task/language combination where it performed the best.

Table 4.10: Best WE models used in each dataset/task/language combination

Word Embedding Model (Mod.-Dim.-Win.)[a]	% of Da/Ta/Ln.[b]	Dataset	Task-Lang.
BERT-768-	16.67%	2014-SM[c]	G[d]-EN, A[e]-EN,
		2014-TW[f]	G-EN, A-EN,
		2014-RW[g]	G-EN, A-EN
		2015	G-EN, CONS[h]-EN
		2017	G-EN
		2018	G-EN
fastText-300-30	10%	2014-SM	G-SP
		2014-BL[i]	G-SP, A-EN
		2015	EXTR[j]-EN, STBL[k]-DU
		2017	V[l]-PT
fastText-300-40	6.67%	2015	AGRE[m]-DU, CONS-IT
		2016	A-SP
		2017	G-AR
fastText-500-30	6.67%	2015	A-SP, AGRE-EN,
		2017	G-SP
		2018	G-SP
fastText-500-40	11.66%	2015	A-EN, STBL-IT, CONS-DU
		2016	G-EN
		2017	V-EN, V-SP, V-AR
word2vec-300-30	16.67%	2013	G-EN, G-SP, A-EN, A-SP
		2014-BL	A-SP
		2015	EXTR-IT, AGRE-SP, CONS-SP
		2016	G-DU, A-EN
word2vec-300-40	5%	2014-TW	G-SP
		2015	G-SP
		2018	G-AR
word2vec-500-30	13.33%	2014-BL	G-EN
		2014-TW	A-SP
		2015	EXTR-SP, STBL-EN, STBL-SP, AGRE-IT, OPEN[n]-EN
		2018	G-SP
word2vec-500-40	13.33%	2014-SM	A-SP
		2015	G-IT, G-DU, EXTR-DU, OPEN-SP, OPEN-DU, OPEN-IT
		2017	G-PT

[a] Model name - Dimensions - Window
[b] Percentage of Dataset/Task/Language combinations solved by each model
[c] Social Media
[d] Gender
[e] Age
[f] Twitter
[g] Tripadvisor reviews
[h] Conscientiousness personality trait
[i] Blogs
[j] Extraversion personality trait
[k] Stableness/Neuroticism personality trait
[l] Language variety
[m] Agreeableness personality trait
[n] Openness personality trait

4.6 Discussion

We can assume that WEs trained over specific-domain datasets, are useful as building blocks to construct DEs. In this sense, the methodology presented in this chapter although straightforward, is arguably a sound strategy to produce text representations (DEs) for AP tasks. This approach outperformed two baselines, often depicted in literature as solid references in NLP problems. Also, it proved to be a strong competitor against the top-quartile participants in six international competitions on AP challenges (PAN shared tasks 2013-2018). Since the *test* partitions of the datasets in such competitions have recently been released, this study might be one of the first that describes an exhaustive comparison between a proposed method and the official results. Furthermore, from the results attained in this study, it can be concluded that not all terms have the same influence when constructing a DE, it depends mostly on the feature to predict (e.g., gender, age) and the domain-specific dataset. Whereas in some dataset-task combination a rare term is very important, in others a common word tends to have more value. Hence the importance of open lexicons, which are derived from the extraction of latent information in dadasets. This might be a reason why the *rtv* rewards differently same terms across different corpus. The dynamic nature of language accounts for the oscillating meaning of some words through time and context, which explains why some terms are AP relevant in some datasets, whilst in other their quality declines. Moreover, the *rtv* statistic devised in this study, demonstrated that the term-theme relation is also very important for the weighting of WEs.

To the best of my knowledge, an evolutionary approach to compute weighting-schemes to aggregate WEs into DEs in the AP context has never been attempted. In this respect, the main contributions of this work can be summarize as follows:

- From the theoretical point of view, the *rtv* introduced in this study, has shown potential as a practical statistical feature, considering that the GP procedure pondered it as the most likely variable to appear alone in a single equation, then suggesting its usefulness in AP tasks, and potentially in more general NLP problems.

- The practical implication of this work is that, the proposed approach can be deemed as a robust methodology, since it attained competitive results in all datasets where it was tested, achieving always a top-quartile place against SoA competitors. Thus suggesting its utility as a feasible solution to predict characteristics from persons behind a social media account.

At the same time, the proposed method present shortcomings that could be addressed in future investigations. For instance, larger fine-tuned BERT models could be employed to produce better customized contextual WEs. In addition, the GP strategy could also be enriched by including even more latent statistics as variables. In the end, even using better contextual WEs a DE can encode only so much information, since the order of the terms in a document is discarded by the aggregation policy. To tackle this flaw, a neural network approach was devised, which is the theme of next chapter.

A paper of this chapter was published in the Elsevier *Information Processing & Management* Journal. Indexed by the Journal Citation Reports (Clarivate Analytics, 2019), Impact Factor: 3.892. [86].

Chapter 5

A Wide & Deep Attentive Network for Author Profiling: The Profiler

5.1 Motivation

As discussed in last chapter, the Author Profiling (AP) problem remains unsolved and new approaches are constantly being evaluated. In the Natural Language Processing (NLP) field the State of the Art (SoA) is frequently updated with novel techniques. Take for instance the case of Recurrent Neural Networks (RNN). Up until just a couple of years, this type of deep neural architectures were the SoA for addressing NLP related problems. However in the last few years this networks have steadily been displaced by new architectures, namely the *transformer*. So it has become customary to try this new approach in areas where it has not been tested before. Nonetheless, there must be a clear idea and justification behind the attempt to use it in different contexts. The *transformer* has been already evaluated with extraordinary results in several tasks such as *translation*, *question answering* and *automatic text generation*. As of now, it has not been assessed in AP tasks. The usage of a SoA methodology where it has never been tried before constitutes innovation. However, the proposal of a novel technique with scientific foundations require a more original approach. Employing the *transformer* as groundwork it is hypothesized that a new and expressly AP designed *transformer* based methodology could deliver competitive and even SoA achievements.

The main mechanism behind the *transformer* is the self attention module. This component provides a careful scrutiny among words in a document (i.e., sentence, paragraph). This process supplies terms with context, which has been found to be of great benefit in several NLP tasks. Later in the chapter, we will discover that paying attention to words in documents might not be enough for AP tasks. So a new architecture is envisioned that takes advantage of the attentive *transformer*, whilst simultaneously learns AP specifically designed features. Literature suggests that "personal"and "thematic" information is useful in the profiling of authors. Then it is theorized that a jointly effort that pays attention to context of terms and also grasps "personal"and "thematic" features would attain positive results.

Moreover, by the end of last chapter the shortcomings of the method there presented were clearly stated. So another main motivation behind the architecture introduced in this chapter is to address the AP riddle from a DL perspective. As with the first method, some of the datasets employed before are used again, particularly those where Deep Learning (DL) based methods were utilized the most, for comparison purposes.

5.2 Related Work

Historically, in the *Uncovering Plagiarism, Authorship and Social Software Misuse* (PAN) scientific event the winning teams in years 2013 to 2019 have used well known strategies that could be considered "traditional", such as *Bag of Words* (BoW), *Second Order Attributes* (SOA), *Latent Semantic Analysis* (LSA) and *tf-idf weighted character and word n-grams* [42, 43, 44, 45, 46, 47, 48]. However In recent years DL approaches have proved their quality in NLP endeavours, this is why the amount of works using DL strategies has recently increased in NLP shared tasks. Take for instance the work of González et al. in [75]. Their research attempted the prediction of irony in Twitter posts using a *transformer* based architecture (similar to BERT [64]) to contextualize in-domain word2vec embeddings. What is interesting of their approach, is that they could examine the multi-head attention module of their architecture to asses how irony is detected within the text. Their work was tested at the *Irony Detection in Spanish Variants* (IroSvA) task at the *Iberian Languages Evaluation Forum* IberLEF 2019 and *Irony Detection in English Tweets* task at the *International Workshop on Semantic Evaluation* (SemEval 2018), achieving 1st and

2nd place respectively.

However, for the PAN AP tasks the results attained by this new wave of techniques are yet to consistently achieve top places. Nonetheless, they have been steadily gaining ground. The PAN datasets that will be treated in this chapter are those from the 2017, 2018 and 2019 editions, since it is in these years the quantity of DL methods has incremented, thus an interesting comparison could be made. To exemplify this trend, some works done by participants in the 2017 to 2019 editions at PAN are described. First we have the work of Franco-Salvador et al. in [106], where the authors used a Deep Averaging Network (DAN) fed with in-domain trained Word Embeddings (WE) using the fastText algorithm [85]. Their effort attained 10th place out of 22 participants. Next we have the approach of Kodiyan et al. in [107], in this case the authors attempted the profiling of authors by using Bidirectional RNNs, using Gated Recurrent Unit (GRU) cells and an attention mechanism. Their strategy achieved 12th place at the same competition. Then, there is the work of Miura et al. in [50], where they addressed AP by the use of a sophisticated DL architecture containing WEs, RNN and Convolutional Neural Network (CNN) layers and attention mechanisms. This SoA methodology achieved 4th place overall. Even though it did not obtained a top rated result, it can be seen that showed a competitive performance, which gives hope for DL approaches to start succeeding at these events. In the work of Nils Schaetti in [108], the competitor approached the task by enrolling two models, first a traditional scheme that used a *tf-idf* strategy to encode features, and the second model employed a CNN. The first model was used to predict language variety and the second to forecast gender. The combined effort achieved 11th place out of 22 participants. Finally, the work of Sierra et al. in [109] used two CNN models to separately address language variety and gender predictions. This time the authors obtained the 8th place overall.

For the 2018 edition of the AP task at PAN, DL approaches were tried again. In [110], Bayot et al. used WEs trained over Wikipedia texts, using the Skipgram variant of the word2vec algorithm. Then these WEs were fed to a Long Short-Term Memory (LSTM) RNN to predict the gender of Twitter posts. They achieved the 18th place out of 23 competitors. Besides the AP task, the PAN event also hosts several other challenges that are closely related to AP, that is the case of the *Style Change Detection* (SCD) problem (i.e., to determine if a text is written by one or many authors). The work of Hosseinia et al. in [51] is worthy of mention since attained 2nd place overall at this competition. The authors approached this task by using two parellel LSTM RNNs with attention mechanisms. Back to the AP problem, Martinc et al. employed in [111] a multi-modal approach, that is, using both text as well as image posts from users to attempt the prediction of gender. For the textual part of the model they used a CNN that was fed *tf-idf* statistics of words, along with character and word embeddings. For the latter, fastText pre-trained WEs were employed. For image prediction they used a pre-trained CNN-VGG-16 type architecture. The combined multi-modal methodology obtained the 8th place out of 23 participants. Next, Nils Schaetti also employed a multi-modal approach in [112], using a character-level CNN for the text part and the pre-trained ResNet18 (as in 18 layers) for images. This technique attained 16th place. Finally, Sezerer et al. achieved the 20th place out of the 23 contestants fusing a CNN with attention mechanisms [113].

Lastly, in the 2019 edition at PAN the objective was to predict whether an author was a *bot* or a *human*, and also the *gender* (e.g., *bot*, *female*, *male*). Again, more DL strategies were put to the test, that is the case of Polignano et al. in [52], they used a CNN approach, and was the DL method best ranked achieving the 11th place overall. In addition, but with more limited success several DL techniques were also evaluated, using various forms of architectures, such as a merged strategy of RNNs with CNNs, RNNs with hierarchical attention and a LSTM RNN with a voting strategy [114, 115, 116, 117, 118, 119].

It can be acknowledged that with a few exceptions, most of the novel DL approaches employed at different AP shared tasks achieved sub-optimal results, and the top-quartile places remain under control of "traditional" methodologies. Nonetheless, the number of DL techniques tested at these events is expected to increase in the next years. Whether these type of methodologies could outperform "classic" methods or not, is yet to be seen.

5.3 Methodology

5.3.1 Background

To address the problem of AP using the PAN datasets, it is common to employ a "traditional" ML approach, since these methods had historically achieved the best results. One reason for this could be the nature of the training data, being relatively small for DL standards. The method introduced in this chapter focuses on a DL architecture that performs competitively even with limited size datasets. This novel architecture called the *Profiler* is based on the self-attention mechanism introduced by the *transformer* network proposed by Vaswani et al. in [68], and on the the *Wide & Deep* architecture conceived by Cheng et al. in [13]. The *transformer* has become the "go-to" DL method for NLP endeavours in the last months, even beginning to make RNNs a little obsolete. This methodology has shown its quality mainly in translation tasks and the pre-training of Language Models (LM), capable of also achieving SoA results in language understanding tasks, such as *General Language Understanding Evaluation* (GLUE) comprised of 9 tasks, the *Stanford Question Answering Dataset* (SQuAD) and *Situations With Adversarial Generations* (SWAG) among others. To the best of my knowledge, the *transformer* has not yet been used in the AP problem, nonetheless it has been employed in NLP related tasks such as *irony detection* by González et al. in [75].

Initially, to address the AP problem a *transformer* based architecture was implemented to asses the reach of "only" attention mechanisms. This methodology is comprised of a single head self-attention module for contextualizing Word Embeddings (WE). This model will be referred to as the *Transformer Based Contextualizer* (TB-C) for the rest of this chapter. See *(b)* model in Fig. 5.1

On the other hand, another *transformer* based architecture is introduced, which is the main contribution of this chapter. The *Profiler* is a *transformer* based network with *wide* and *deep* branches specially designed for AP tasks. From here on it will be known in figures and tables as the *Wide & Deep Transformer* (WD-T). See *(a)* model in Fig. 5.1. Furthermore, to establish a frame of reference to asses their performance, they were compared with official results from the AP shared tasks at PAN 2017 to 2019.

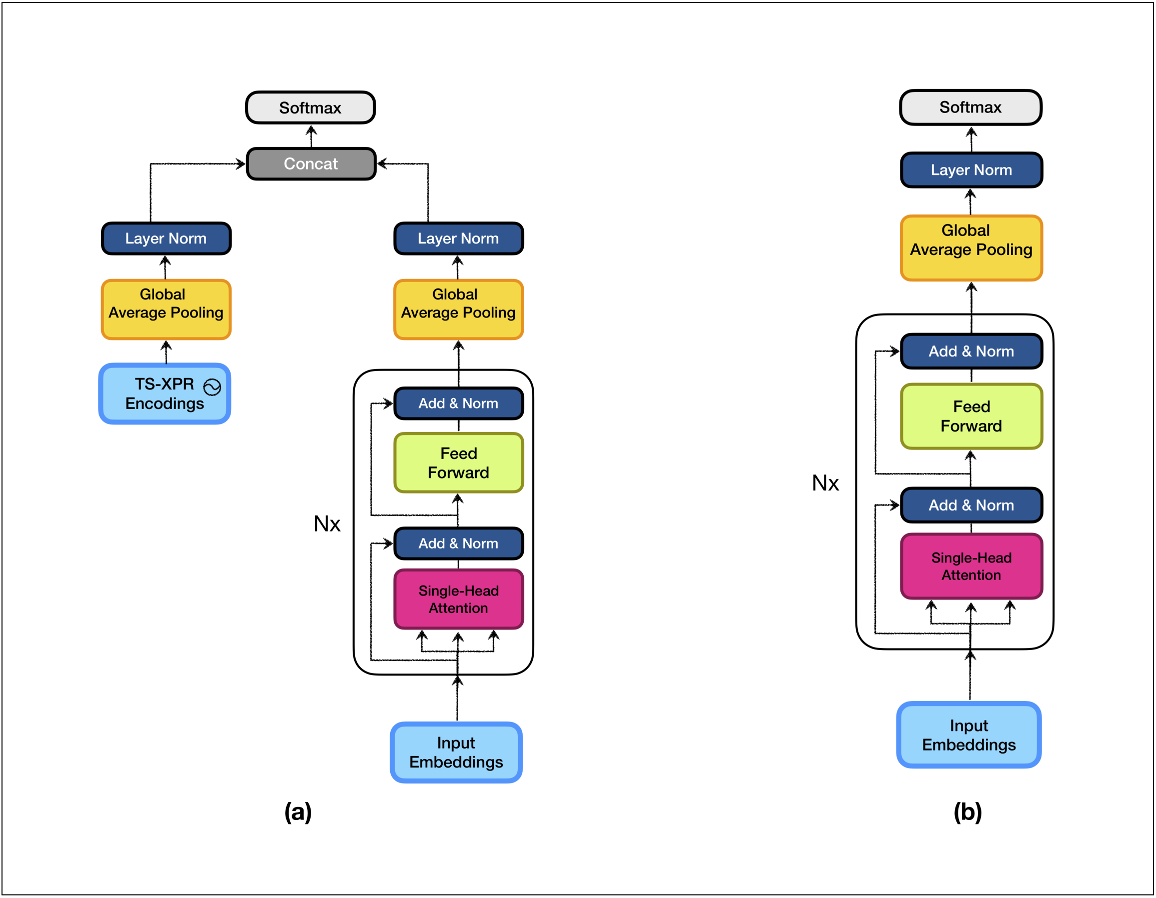


Fig. 5.1. The Profiler on the left (a) (WD-T) and the Transformer based contextualizer (b) (TB-C) for author profiling

5.3.2 The proposed architecture

Both, the WD-T and the B-TC are based on the *encoder* module of the original *Transformer*, since the task at hand is classification, we can dispense of the *decoder* module. As can be seen in Fig. 5.1, both approaches shared the same encoder model, since it is in here where the self-attention mechanism is implemented. As explained in **Chapter 4**, WEs are non-contextual word vectors, and since context is important in NLP tasks, the self-attention structure "contextualizes" them. First, the encoder module is provided with WEs produced with the *word2vec* skipgram algorithm, also explained in last chapter. Other WEs algorithms were explored such as *fastText*, along with multiple combinations of hyper-parameters, as was done also in last chapter. Ultimately the best WE technique and hyper-parameter blend was *word2vec* skipgram with a *window* of 30 and 300 *dimensions*. These WEs were produced using each *train* partition of the 2017, 2018 and 2019 datasets for English and Spanish languages, thus producing six different sets of WEs. These datasets are composed of 100 Twitter posts from each user. A *document* (d), which will be the term used from here on to depict the text produced by a user, is comprised as the sequential concatenation of those 100 posts from each user, thus having variable length documents across the dataset. Once the WEs were obtained, the *document* representations to feed the encoder are composed as follows:

- Since the encoder needs fixed length sentences, the maximum *document* length both in *train* and *test* partitions is computed for each dataset (year/language combination). See statistics of the datasets in Table 5.1.
- A *document* is formed as an array of WEs representing each word in it, with a length of the maximum *document* found in the current dataset, padded from left to right with vectors of zeros (0's) with the same dimension of the WEs (300).

The whole travel of the WEs through the *encoder*, up to before the final softmax function that predicts the output is explained next:

1.- The *document* representations are fed to the self-attention module. This module is called *multi-head* attention in the original *transformer*, because is comprised of up to 8 identical self-attentive mechanisms (heads), each one "pays" attention to different aspects of the words (e.g. tense of the verbs). For the AP task in this work an ablation process showed that a single-head self-attention delivered the best results, hence the attention mechanism proposed in this thesis contains only one head. The self-attention process can be summarized in Eq. 5.1. Each WE is multiplied by three different matrices: W^Q, W^K and W^V (initialized before training with the *glorot_uniform* method), thus producing the vectors: $\overrightarrow{Q}$, $\overrightarrow{K}$ and $\overrightarrow{V}$, as shown in Fig. 5.2. In a nutshell what the attention mechanism do is apply the *dot-product* attention between the words (WEs) in each document but by scaling it by the factor $\frac{1}{\sqrt{d_k}}$, where d_k is the dimension of the attention head, this is done to attenuate the effect of extremely small gradients. Note that attention vectors $\overrightarrow{Q}$, $\overrightarrow{K}$ and $\overrightarrow{V}$ have lower dimensions than the WEs (64 as proposed by the original *transformer* paper).

This dimension reduction is only done to make the estimate of the attention constant and more memory efficient. These vectors allow to calculate a score of "how much" attention to pay in each word in the document. The softmax function normalizes the scores and establishes an attentive amount of each word at its position with respect to the rest of the words. Next, the value vector of each WE is multiply by the softmax result, so the "less" important words can be diminished and also the opposite is true for the "more" important terms. Finally, all the weighted (by the softmax) value vectors are added up to produce the Z vector, which is the original WE with its "attention" embedded (contextualized). The computation of the self-attention phase is quadratic in complexity ($O(n^2)$), and also occurs entirely in memory, thus the use of a proper Graphics Processing Unit (GPU) is almost mandatory even for relatively small sized datasets. Also it is important to mention that the operations over vectors depicted in Fig. 5.2 are done at matrix level in the actual implementation for efficiency purposes.

Table 5.1: Amount of samples in each dataset. Also, the average length of train and test documents is included. The length of a document is measured in number of tokens (words) in each sentence. The standard deviation is included for reference.

		Documents			***Train length stats***		***Test length stats***	
Dataset Lang.	***Classes***	***train***	***test***	***total***	***Avg.***	***Max.***	***Avg.***	***Max.***
2019 EN[a]	T[b]=2, G[c]=3	4120	2640	6760	1948.02	5533	1950.19	6721
SP[d]	T=2, G=3	3000	1800	4800	1784.27	4966	1779.32	5155
2018 EN	G=2	3000	1900	4900	1597.24	2611	1595.17	2630
SP		3000	2200	5200	1532.18	2861	1540.45	2749
2017 EN	G=2, V[e]=6	3600	2400	6000	1592.89	3258	1589.65	2707
SP	G=2, V=7	4200	2800	7000	1517.35	2861	1528.60	2749

[a]English
[b]Type
[c]Gender
[d]Spanish
[e]Variety

$$\text{Attention}(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{5.1}$$

2.- Next, the output of the self-attention module is pass through an *Add & Norm* stage, which is described in Fig. 5.3. What this phase does is take the original WE vectors and add them to the Z vectors in their respective positions, then the produced

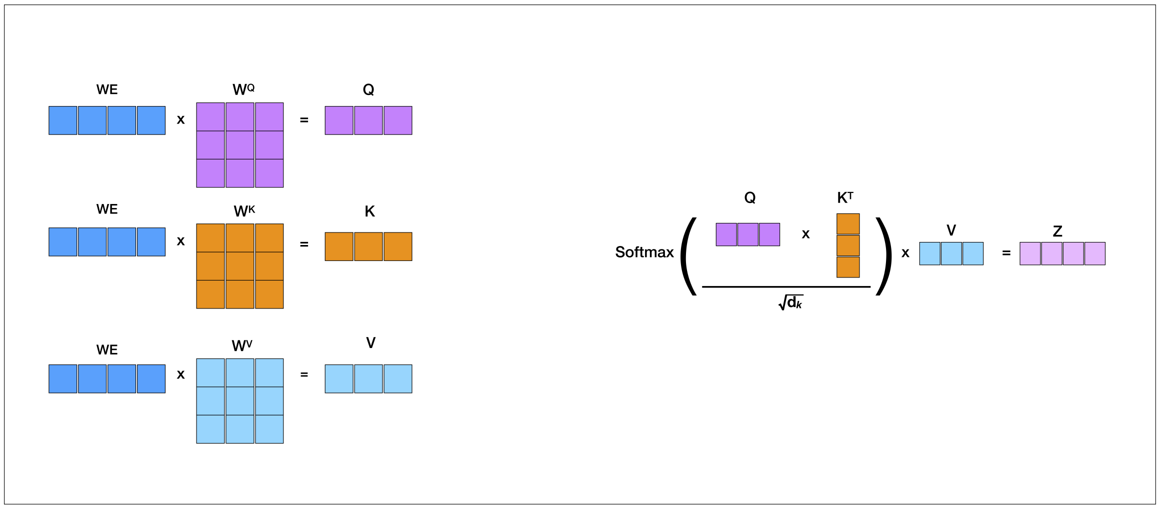


Fig. 5.2. The Self-Attention mechanism.

added vectors are processed with a *Layer Normalization* algorithm, which opposite to for example a mini-batch, normalizes the values feature-wise instead of by samples.

3.- Then, the output of this *Add & Norm* module is fed to a Feed-Forward Neural Network (FFNN), which also is in turn passed to another *Add & Norm* module with the residual connections of last stage. See Fig. 5.1.

Steps one to three comprises the *encoder*. Which can be stacked Nx times, as also seen in Fig. 5.1. Since the amount of samples in the datasets employed in the experiments of the present work is limited, the best results were attained with only one encoder ($N = 1$). But it is worth explaining how the stacked encoders work. If several structures were to be stacked, only the first encoder would receive the WEs of the *document*, then the output of the first encoder is fed to the next one and so on. Since in this case there is only one encoder, its output is passed directly to a *Global Pooling Average* stage, which summarizes the Z attentive vectors dimension wise for better performance. Finally, right before the softmax function used for classification, a *layer normalization* module processes the pooled output.

For the TB-C model, this is the complete methodology. The softmax activation function is finally employed to calculate the probability of each class, depending on the task (see Table 5.1).

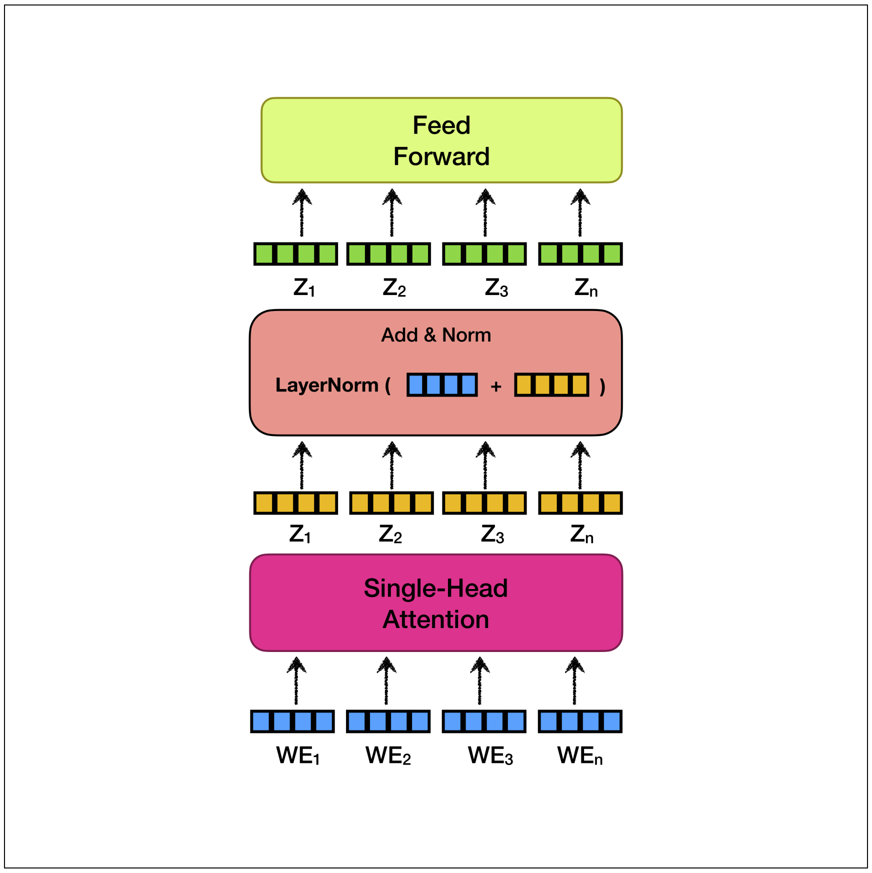


Fig. 5.3. The Add and Layer normalization phase after self attention

5.3.2.1 The Profiler aka the Wide & Deep Transformer (WD-T)

However as previously established, for AP tasks "attention" might not be "all you need". This is why *the Profiler* (WD-T) is introduced. Same as with the TB-C, the WD-T employs the same encoder architecture, which is the *deep* part. For the *wide* element, the method needs to incorporate known useful features for the AP problem. As stated by Pennebaker in [104], certain words have the power to identify characteristics of authors of texts. For instance in such work, it is noted that *men* tend to employ articles at a higher rate than *women*. At the same time *women* use personal pronouns such as 'I', 'me', and 'my' more frequently, as well as verbs. Thus, as proposed by Pennebaker and also by Ortega-Mendoza et al. in [41], personal information is very useful for the profiling of individuals. The WD-T incorporates this information into the network through the *wide* branch, so the model can predict more accurately the type of author (e.g., gender).

In this regard, the addition of the *wide* element is of great importance for the general performance of the WD-T. Without it (i.e., only the TB-C model) as the reader will notice in the 5.5 section, the overall efficiency is significantly reduced. In the following lines, the rationale behind the *wide* component is explained.

Inspired by the work of Ortega-Mendoza et al. in [41], a novel measurement was devised to represent the style and personal information of an author. This value is called *Term Specificity-Exponential Personal Reward* (TS-XPR). To compose the TS-XPR, first we need to compute a precision (ρ) and recall (τ) values, as proposed by Ortega-Mendoza et al., in turn also inspired by information retrieval evaluation metrics transferred to the context of personal information. To accomplish this we need to identify those sentences which include personal information. As the lector can remember, each user's document is formed by the concatenation of exactly 100 tweets, hence each post will be treated as a *sentence*. In this way we can divide a document d_j (being j the j-esim user), in two sets of sentences: P_j and S_j, where P is the set of all posts containing personal information, and S all posts in total. In order to classify a post as belonging to the set P, it is necessary to verify if any of the next personal pronouns appears in it: "i", "we", "me", "us", "my", "mine", "our", "ours", "myself", "ourselves" for English, and "yo", "me", "mi", "conmigo", "nosotros", "nos", "nosotras" for Spanish.

Therefore, we have Eq. 5.2 (ρ), which represent the amount of personal information contained in a word, that is to say the relation of the number of personal posts to all posts. In addition, Eq. 5.3 (τ) reflects the fraction of the set of personal sentences (P) the term t_i appears on, being i the i-esim word in the document d_j.

$$\rho_{(t_i,d_j)} = \frac{\#(t_i, P_j)}{\#(t_i, S_j)} \tag{5.2}$$

$$\tau_{(t_i,d_j)} = \frac{\#(t_i, P_j)}{\#(P_j)} \tag{5.3}$$

After computing ρ and τ, an F-score like metric can be calculated in order to produce a more stable measure of *personal information*. Eq. 5.4 shows how to compute this score.

$$\mathrm{F}_{(t_i,d_j)} = 2 \cdot \frac{\rho_{(t_i,d_j)} \cdot \tau_{(t_i,d_j)}}{\rho_{(t_i,d_j)} + \tau_{(t_i,d_j)}} \tag{5.4}$$

Moreover, this information can now be used to weight terms in a document, whether if appears on a personal sentence (P) or not. Ortega-Mendoza et al. also proposed a weighting scheme based on the F measure. The rationale behind it is that a term weight value begins from the overall frequency of the term, then it is exponentially rewarded only if appears in personal sentences, which will not occur if it surfaces in non-personal posts. In addition, an offset value that describes the usage of words across documents is added to the weight of the term, thus reflecting not only the *"personal"* information magnitude it contains, but also the importance of such depending on the user-specific usage of the word. Eq. 5.5 describes this whole idea called the Term Specificity-Exponential Personal Reward (TS-XPR).

$$\text{TS-XPR}_{(t_i,d_j)} = \log\left(\frac{\#(d_j)}{1 + \#(t_i, d_j)}\right) + 1 + \left(\sqrt{\frac{\#(t_i, d_j)}{\#(d_j)}}\right)^{1-\mathrm{F}_{(t_i,d_j)}} \tag{5.5}$$

As can be seen, the first term of the equation is the commonly known *inverse document frequency* (idf), computed locally for each user's posts to offset the importance of a word depending on its usage by each author. The second term of the formula is the weighting scheme proposed in [41].

The original *transformer* was devised mainly for translation purposes, hence the *decoder* module was also needed. In addition, in order to learn the context of words, meaning the significance of the place a term occupies in a sentence or document, said *transformer* needed to learn the position of words, which a RNN implicitly learns by sequentially processing the inputs. A *transformer* does this in a more efficient manner by adding a Positional Encoding (PE) to the input WEs. Such PEs represent the order of words encoded simultaneously in a *sinusoidal* and a *cosine* signal, with the added advantage that such functions are infinite in time, hence not limiting the length of the sentence being encoded. For the WD-T (as well as the BT-C), such PEs were actually detrimental, thus it can be assumed that the actual position of words is irrelevant in the AP task, focusing the attention only in the neighboring words to establish context.

Up to this point, the set of TS-XPR values for each term in a user's document needed to be incorporated into the network. Several approaches were tried, such as adding or multiply them to the input WEs. The idea was to influence the self-attention mechanism so it paid more attention to "personal" and "thematic" terms. Needless to say that it did not work as expected. So another method was attempted. Following the original idea of Vaswani et al. in [68] to compute the PEs, the same method was used to encode the TS-XPR values into a *sinusoidal* and a *cosine* signal to obtain encodings or vectors instead of single values. Eq. 5.6, is used to convert the TS-XPR values into encodings of the same dimension of the WEs (300), which are now called Term Specificity Personal Encodings (TSPE). The idea is to produce embeddings of $\#dim$ dimensions by computing the *sin* and *cos* of the TS-XPR value divided by the factor $\frac{1}{10000^{(2k/\#dim)}}$, where k is the current iteration starting from $n = 1$ until the number of dimensions is reached. If the iteration k is even, the *sin* function is invoked, otherwise the *cos* function is calculated, thus producing a vector of $\#dim = 300$ alternating the values with *sin* and *cos* functions depending if the dimension is even or odd (See Eq. 5.7).

$$\text{TSPE}_{(t_i,d_j)} = \begin{cases} Sin\left(\frac{TS\text{-}XPR_{(t_i,d_j)}}{10000^{(2k/\#dim)}}\right) & \text{if } n = 2k \\ Cos\left(\frac{TS\text{-}XPR_{(t_i,d_j)}}{10000^{(2k/\#dim)}}\right) & \text{if } n = 2k+1 \end{cases} \tag{5.6}$$

$$[sin(W_{(2_k)}), cos(W_{(2_k+1)}), ...] \tag{5.7}$$

where $W = \frac{TS\text{-}XPR_{(t_i,d_j)}}{10000^{(2k/\#dim)}}$

Once the TS-XPR values were converted into TSPEs, the most logical option was to add or concatenate them to the input WEs, just like the PEs in the original *transformer*. This also came to no fruition. The self-attention mechanism was again unable to grasp the intended encoded information. One reason could be that the TSPEs subtly conceal the needed information, so a novel methodology was devised to integrate the TSPEs into the network, so they could be more clearly fathomed.

The manner the TSPEs were incorporated into the network is inspired by the work of Chen et al. in [13], which introduces *wide & deep* architectures that could somehow merge the generalization power of deep networks and the task-specific feature engineering of linear models. The original *wide & deep* network simply utilized a general linear model of the form $y = W^T x + b$ for the *wide* features, and a FFNN as the *deep* component. Then the *wide* and *deep* components are integrated using a weighted sum of their outputs, for finally establishing a joint training stage using a logistic loss function for classification, which is common for both components. On the other hand, the WD-T model proposed in this thesis is shown in Fig. 5.1 left graph (a). It can be observed that the TSPEs, which represent the *wide* element, are processed parallel to the encoder module. These TSPEs are also passed through a *Global Average Pooling* and a *layer normalization* stage, then they are concatenated to the vectors outputted by the encoder, right before the final softmax function used for classification. The rationale behind this architecture relies in feeding the network with both, the WEs that are in turn processed with the self-attention mechanism (i.e., *deep* component) and the *wide* feature vectors that encode the *stylistic* and *personal* information of the words (TSPEs). The same structure for composing the document representation of WEs to be fed to the encoder is used to construct an array of the *wide* features, but instead of the WEs the TSPEs are put in the respective place the term occupies in the document. The *wide* component of the network was able to learn the *wide* features simultaneously with those (i.e., *deep* element) discovered by the self-attention module, thus producing a joint *wide & deep* learning by means of sharing the same loss function, in this case *categorical-crossentropy.* It is worth mentioning that the *backpropagation* of the error is done parallel in the *wide* and the *deep* branches.

5.4 Experimental Setup

As mentioned before, the experiments addressed in this chapter tackle the AP problem in datasets of the PAN initiative of years 2017 to 2019. In table 5.1 the statistics of such datasets describe the size of the *train* and *test* sets for each year, also it is important to note that the maximum length of the documents is shown. This is relevant because the input of the proposed models is fixed in length, so the maximum length is taken into account to determine the size of the padded input for the models. In Fig.

5.4 can be seen that for the 2019 dataset, the maximum length of the concatenated documents is the largest of all three years. This might be caused in part for the task of detecting bots from humans. Most of the posts produced by bots are very large due to the excessive use of hashtags and re-tweets. In addition, due to restrictions in memory for processing the self-attention module, the 2019 dataset was truncated to a maximum of 3500 tokens (WEs) of length.

For the pre-processing of the datasets a typical approach was employed. Words preceded by a "#" sign were renamed with the label "hashtag". Likewise, words preceded with "@" and "www" were replaced with "user" and "url" respectively. In addition all words were lower-cased, and for the Spanish language the accent sign was removed. A lemmatization process was also performed, using *The Natural Language Toolkit* (NLTK) python library for English [120], and the *Stanza python library from the Stanford NLP Group* for the Spanish language [121].

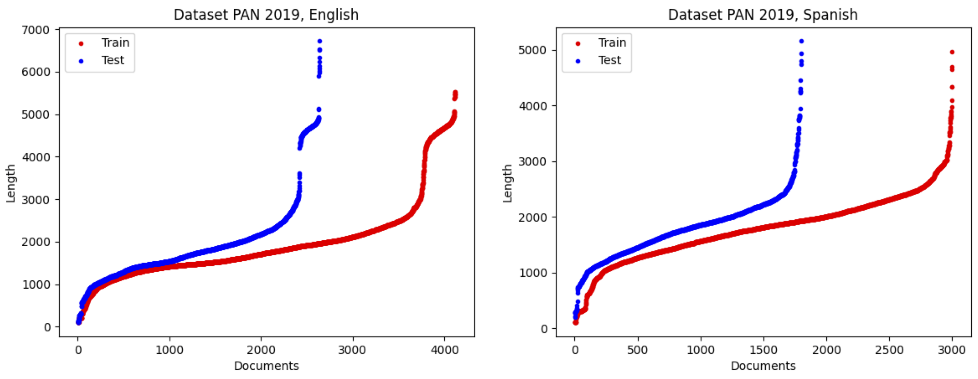


Fig. 5.4. *train* and *test* samples for the 2019 dataset.

For the hyper-parameters of the *encoder* most of the default parameters proposed by Vaswani et al. in [68] were used. The *dimension* size of the WEs was set to *300* with a training window value of *30*. The number of attention *heads* departing from the original transformer (*8 heads*) was set to *1* with a size of *64*. The batch size for the training and evaluation was set to *32*, and the length of the input document is the maximum length of documents for each year's dataset (e.g., *2630* for 2018-English, *2861* for 2017-Spanish). The *Adam* optimizer for the loss function was used with hyper-parameters *learning_rate*=0.001, *beta_1*=0.9, *beta_2*=0.999. In the original *transformer* architecture the *Noam learning rate schedule* is used to alter the *learning_rate* between batches of training, however a fixed value delivered better results than using the scheduler. Furthermore, as already described only *1* encoder layer was used, since this configuration delivered better results. It is worth mentioning that an ablation process was performed to determine the optimal number of stacked *encoders* as well as the number of *heads* in the self-attention module and the optimal *learning_rate* scheduler. Finally, a patience of *30* epochs was established as early stopping policy.

5.5 Results

As already discussed, normally the AP shared tasks of the PAN initiative are attained by "traditional" ML approaches. However, in the last couple of years more DL approaches have been tried but with limited achievements overall. It can be speculated that one of the reasons for this is the general size of the datasets, which might be considered "small" for DL standards. In this chapter two DL models are proposed to tackle AP tasks in the datasets described in table 5.1. The results achieved by these models were positioned in top-quartile places of official results in the competitions. Noteworthy is the performance achieved by of the central model introduced in this work (WD-T), which also outperformed all participant DL methodologies. Fig. 5.5 shows the performance of the WD-T and the TB-C in five shared AP tasks.

When compared only with DL proposals in each year's competition, it can be seen that the main contribution of this work: *The Profiler* or *Wide & Deep Transformer* (WD-T), outperformed every other DL method. Take for instance the results shown in Table 5.2, the WD-T model clearly surpassed the results of all DL proposals in the 2017 gender task.

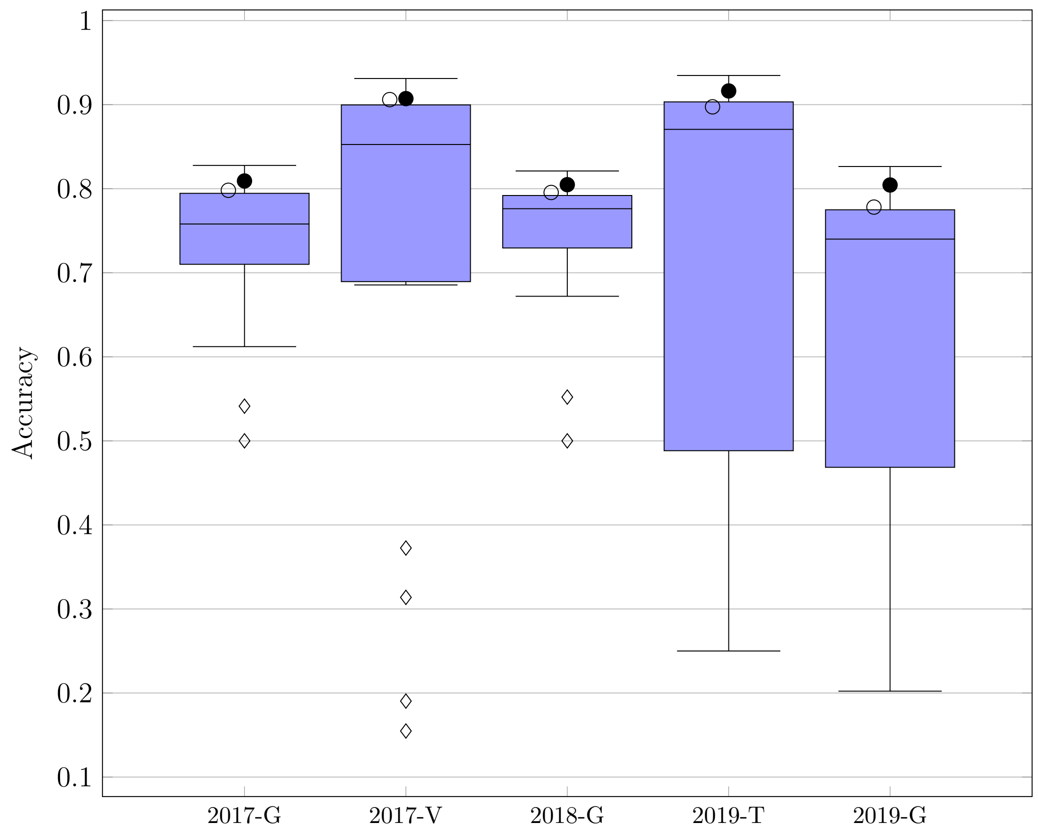


Fig. 5.5. Boxplots depicting the performance of the WD-T model and the TB-C vs all competitors at PAN (2017-2019). A black filled circle represents the WD-T results, and a blank circle depicts the TB-C model. The shown results are the average of English and Spanish languages for each dataset/task. Diamonds portrait the outliers.

Table 5.2: Accuracy of the WD-T and TB-C models & best DL proposals at 2017 gender task.

Ranking	Team	English	Spanish	Average
1	WD-T	**0.8133**	0.8050	**0.8092**
2	Miura et al. (RNN+CNN+Attention)	0.8046	**0.8118**	0.8082
3	TB-C	0.8041	0.7921	0.7981
4	Franco-Salvador et al. (DAN[a])	0.7958	0.7721	0.7840
5	Sierra et al. (CNN)	0.7821	0.7700	0.7761

[a]Deep Averaging Networks

Similarly, for the 2017 language variety task the WD-T attained the best DL results. It is worth noting though, that both the WD-T and TB-C models achieved very similar results, meaning that the WD-T could not stand out clearly against the TB-C model. One possible explanation could be the fact that the task at hand (language variety) do not require at the same level the *personal* and *thematic* information of the *wide* element. Thus it can be assumed that *personal* or *thematic* data is not as relevant for identifying the language variety as for predicting the gender of an author. Table 5.3 shows this results.

Table 5.3: Accuracy of the WD-T and TB-C models & best DL proposals at 2017 language variety.

Ranking	Team	English	Spanish	Average
1	WD-T	0.8683	**0.9461**	**0.9072**
2	TB-C	0.8662	0.9457	0.9060
3	Miura et al. (RNN+CNN+Attention)	**0.8717**	0.9271	0.8994
4	Sierra et al. (CNN)	0.8392	0.9450	0.8921
5	Franco-Salvador et al. (DANetworks)	0.7588	0.9000	0.8294

In 2018, the number of DL proposals incremented. In this year's shared task the objective was to predict only the gender of authors, but using both textual and image features. In table 5.4 it can be observed again that the WD-T clearly outperforms the rest of the DL participants, and it is more separated from the TB-C, which demonstrates *personal* features evidently affect in a positive way the outcome of the model. It is important to make it clear that both the WD-T and the TB-C used only textual features for this task.

Table 5.4: Accuracy of the WD-T and TB-C models & best DL proposals at 2018 gender task.

Ranking	Team	English	Spanish	Average
1	WD-T	**0.8178**	0.7918	**0.8048**
2	Kosse et al. (FFNN)	0.8074	0.7918	0.7996
3	Veenhoven et al. (Bi-LSTM with attention)	0.7926	**0.8036**	0.7981
4	TB-C	0.8105	0.7805	0.7955
5	Sierra-Loaiza & González (CNN)	0.8011	0.7827	0.7919
6	Takahashi et al. (RNN+CNN)	0.7968	0.7864	0.7916
7	Martinc et al (CNN+fastText)	0.7900	0.7782	0.7841
8	Aragon & Lopez (CNN+WEs)	0.7963	0.7686	0.7825
9	Schaetti (CNN + resnet18)	0.7711	0.7359	0.7535
10	Sezerer et al. (CNN+attention)	0.7495	0.6655	0.7075
11	Raiyani et al. (FFNN+fastText)	0.7279	0.6436	0.6858

Furthermore, in the 2019 shared task there were two objectives, first to identify *humans* from *bots*, and then predicting the gender (i.e., *female*, *male*, *bot*). In this competition, the organizers made remarks of the fact that even more DL methodologies were put to the test [122]. They also highlighted the work of Polignano & de Pinto (which uses a CNN) [52], as the best positioned DL approach attaining the 11th place overall. Nonetheless, when we see Tables 5.5 and 5.6, it can be observed that the WD-T outperformed this and the rest of the DL proposals. In addition, if the official results are rearranged to place the WD-T, it would achieve 6th place overall and replaces Polignano & de Pinto's CNN at the 12th place.

The reader might argue that the differences in accuracy are not significant enough. However historically in these competitions the top places are not that far apart from each other, and a gain of just a tenth of a percentage point could be very difficult to attain. This is relevant to fully understand the context of the achievements of the

Table 5.5: Accuracy of the WD-T and TB-C models & best DL proposals at 2019 type task (bot vs human).

Ranking	Team	English	Spanish	Average
1	WD-T	**0.9254**	**0.9194**	**0.9224**
2	Polignano & de Pinto (CNN)	0.9182	0.9156	0.9169
3	TB-C	0.9053	0.8894	0.89735
4	Petrik & Chuda (CNN+RNN)	0.9008	0.8689	0.88485
5	De La Peña & Prieto (FFNN)	0.9045	0.8578	0.88115
6	Bolonyai et al. (RNN+hirerch attn.)	0.9136	0.8389	0.87625
7	Onose et al. (RNN+hirerch attn.)	0.8943	0.8483	0.8713
8	Halvani & Marquardt (FFNN)	0.9159	0.8239	0.8699
9	Zhechev (Voted LSTM)	0.8652	0.8706	0.8679
10	Dias & Paraboni (CNN+RNN)	0.8409	0.8211	0.831
11	Qurdina (CNN)	0.9034	0.0000	0.4517

Table 5.6: Accuracy of the WD-T and TB-C models & best DL proposals at 2019 gender task (bot, female, male).

Ranking	Team	English	Spanish	Average
1	WD-T	0.8189	**0.7844**	**0.8017**
2	Halvani & Marquardt (FFNN)	**0.8273**	0.7378	0.7826
3	Polignano & de Pinto (CNN)	0.7973	0.7417	0.7695
4	TB-C	0.7841	0.7483	0.7662
5	Petrik & Chuda (CNN+RNN)	0.7758	0.7250	0.7504
6	De La Peña & Prieto (FFNN)	0.7898	0.6967	0.7433
7	Zhechev (Voted LSTM)	0.7360	0.7178	0.7269
8	Bolonyai et al. (RNN+hirerch attn.)	0.7572	0.6956	0.7264
9	Onose et al. (RNN+hirerch attn.)	0.7485	0.6711	0.7098
10	Dias & Paraboni (CNN+RNN)	0.5807	0.6467	0.6137

WD-T. Even its sustained top performance against different configuration of deep neural networks is a sign of its robustness.

5.6 Discussion

Why do DL architectures underperform in AP tasks at PAN?. One reason, as already discussed, could be the size of the datasets. An ideal dataset for DL (or any ML for that matter) is rarely found or easily constructed. Thus, DL strategies must adapt to small or "less" than ideal datasets. Another possible explanation for the underachievement of DL methodologies in AP tasks at PAN is that the generalization prowess of DL architectures are limited to contextual features. Whilst the AP task is known to be a complicated problem that requires finer grain features to be grasped. These subtle features might not be picked up by the self-attention mechanism, which makes the wide branch in the WD-T of great importance.

For the AP problem, literature suggests that *stylistic* choices, along with usage of words and *personal* information embedded in texts, could greatly help the profiling of authors, from predicting the *gender* and *age*, to even forecasting their *personality*. In this chapter a novel DL model is proposed: *The Profiler* aka *The Wide & Deep Transformer* (WD-T). This architecture takes into account the attention "paid" to the context of words, while also linearly considers features (*stylistic* and *personal*) engineered expressly for the AP problem. The *wide* element can be substantiated with works in the literature that addresses how words help the profiling of authors, take for instance the work of Pennebaker [104] among others. But how to explain the *deep* element (self-attention) involvement in the matter? It has been made clear that the attention in the encoder can not be intentionally focused onto specific words, and that *dot-product* attention is mainly devised to "contextualize" words. But this is no small action, it is fundamental in NLP tasks, and it has been proved to be of great value in AP endeavours.

In Fig. 5.6 left side, the attention put into a segment of a document from an Australian female can be seen. Note that most words pay attention to the phrase "on my face", specially the word "photo". Also, in the right side of the figure, the attention over a portion of a document from a Colombian male is depicted. In this case, the attention is more disperse, but it can be clearly seen that "camino", "luz", "de", "url" and "yo" (Spanish for "road", "light", "of", "I"), are attended more than any other word. In the naked eye, the relationships between words might not be all clear, specially in the example on the right. This can be explained from the stand point of the length of documents. The original *transformer* was devised with mainly translation tasks in mind, which employs relatively short sentences. The transformer proposed by Vaswani et al. took sentences up to 120 words in length. Furthermore, recent works have tried to increase the attention reach on the dependency of larger sentences, that is the case of the *Transformer-XL*, proposed by Zihang Dai et al.

in [123]. Their work expanded the attention span to 640 words, and even showed acceptable performance with up to 3800 tokens. On the other hand, the role of the self-attention mechanism that contextualize custom pre-trained WEs is not entirely clear in AP tasks with documents of up to 3500 words. Moreover, when the attention is observed on just a small portion of the whole document, the picture could be deceiving. Nonetheless it is evident that the process of contextualizing WEs with self-attention, even in very large documents, plays an important role in the learning mechanism.

Furthermore, as mentioned earlier the length of the input documents for the 2019 dataset must be truncated to 3500 words. The reason was the limited memory of the GPU card employed in the experiments[1]. The lector can appreciate in the footnote that the computational resources employed are of considerable capacity, yet with very large documents even those could be insufficient. One can only wonder whether the WD-T results would be even better should lengthier documents were allowed. The answer could be closer than imagined. In the recent work of Kitaev et al. entitled *Reformer: The Efficient Transformer* [124], they introduced a more efficient architecture that replaced the *dot-product* attention with a *locality-sensitive hashing*, which reduced the computational complexity from $O(n^2)$ to $O(nlogn)$, where n is the length of the document. They attained comparable results to the original *transformer* with less computational resources. So a possible future work envisioned for the WD-T could be an improvement of the current self-attention mechanism with the one mentioned by Kitaev et al. or even a novel one.

In Fig. 5.7 two samples of attention on correctly classified specimens can be seen for the 2019 dataset. On the left side a sample of a *bot* is shown, whereas on the right side a *human-female* is depicted. It is interesting to note that the *bot* sample on the left side, is paid more attention on the terms closer to the word “hashtag”, which is consistent with the idea that *bots* significantly employ more often hashtags and re-tweets. In the example on the right, the document excerpt of the human-female shows that the words “mi”, “yo”, “url” and “gustar” (Spanish words for “my”, “I”, “to like”) are paid more attention than others. This is also consistent with the rationale of women being more self-aware than men [104].

[1]An NVIDIA Titan RTX GPU with 12Gb of memory installed in a server with an Intel Core i9 CPU with 64Gb of RAM

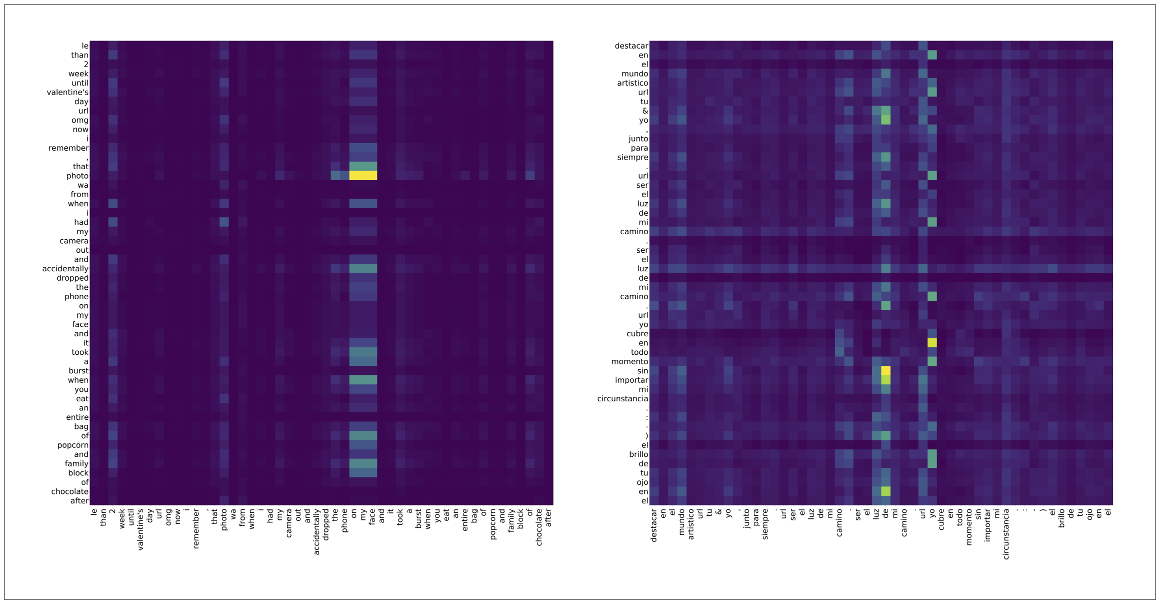


Fig. 5.6. Single-head self-attention on 2017 dataset. Left graph depicts attention in the English/gender task, right graph represents attention on the Spanish/language variety task. A lighter color represents "more" attention.

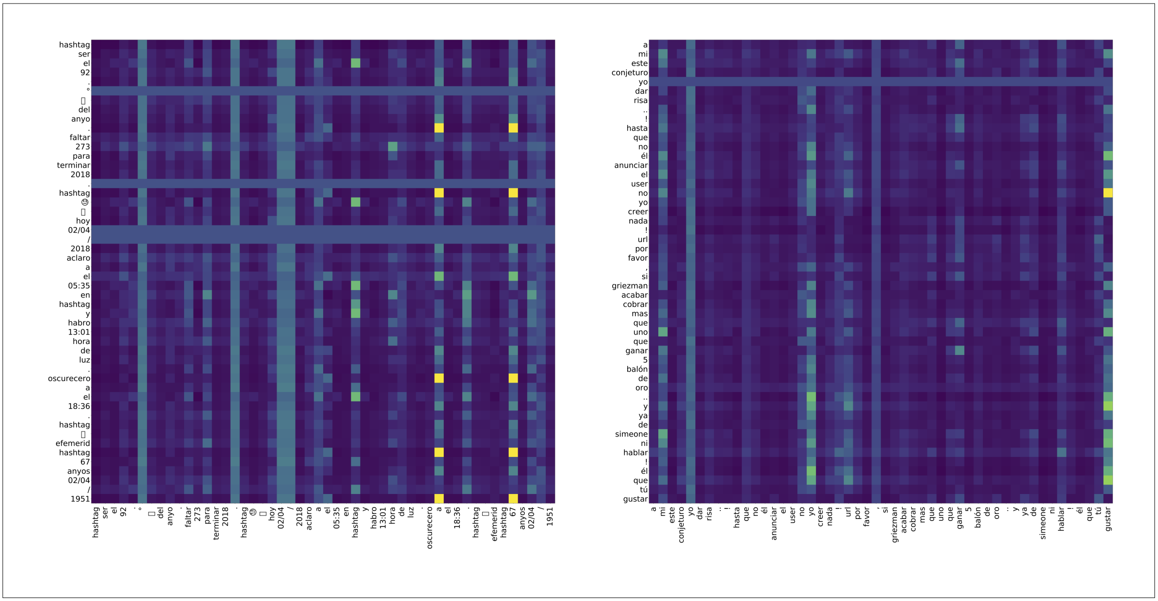


Fig. 5.7. Single-head self-attention on 2019 dataset. Left graph depicts attention in the English/type task (bot vs human), and the right graph represents attention on the Spanish/gender task (human, female, male). A lighter color represents "more" attention.

Finally, in the 2018 sample of a *female* seen in Fig. 5.8. It is interesting to notice that the words “david” and “bowie” are related. Also, the words “book”, “lock” and “chooks” (which is often employed in Australia to refer to *chickens*), are paid attention from many words, specially for “chooks”. Although it is not disclosed if the female is from or resides in Australia, it would be amusing to find out the nationality of the user.

The samples shown in the attention Figs. 5.6, 5.7 and 5.8 are sentences extracted from whole documents (100 concatenate tweets for each user), some of them are comprised of two or three tweets, maybe even topic unrelated. So the attention depicted in the figures might be a misleading window into what the self-attention module is really focusing on for each user. In addition, as it happens with most DL architectures, the interpretation of the inner structures such as the *path lengths* of the neurons or the attentive *dot-product* scores, can only be partially explained, their true meaning or reasoning behind them might remain undiscovered for a while.

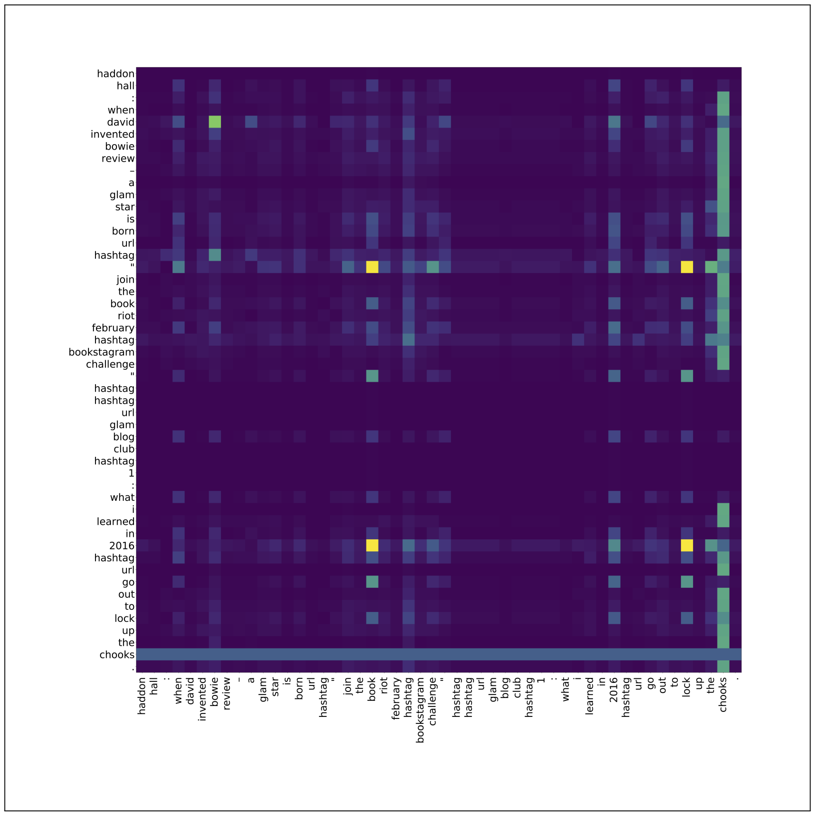
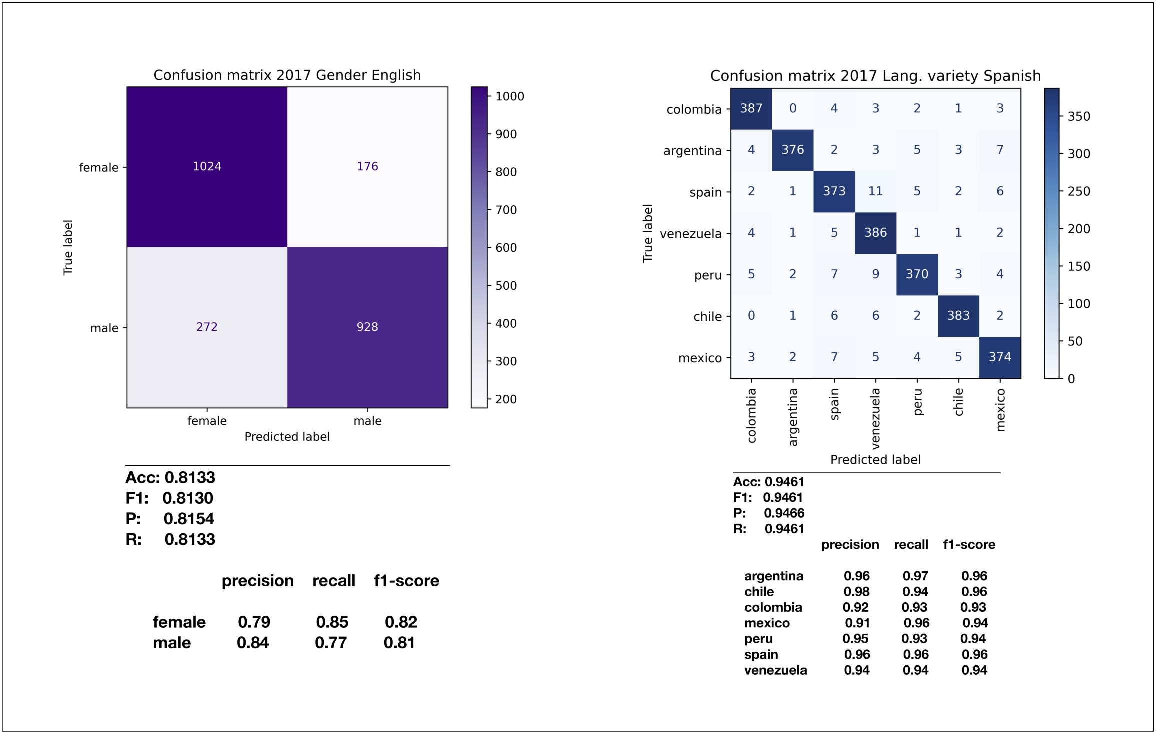


Fig. 5.8. Single-head self-attention on the 2018 English/gender task. A lighter color represents “more” attention.

In Fig. 5.9 the performance of the WD-T on the 2017 dataset is shown. It can be seen on the left side the *gender* prediction task for English and on the right side the *language variety* forecasting for the Spanish language. In the confusion matrix and the summary of the *accuracy*, F_1 *score*, *precision* (p) and *recall* (r), the scores seem very stable, meaning that the F_1 *score* demonstrate the balance between the p and r values, and even very close to the *accuracy* metric. In addition, a similar strong performance is described in Fig. 5.10 for the 2018 and 2019 datasets. On the left side the 2018 *gender* prediction task for English is displayed and on the right side the 2019 type (e.g., "bot", "human") for the Spanish language.

In conclusion, to the best of my knowledge, transformer based models for the AP task have not been employed so far in any shared task competition, or published in literature. The two models here proposed (WD-T & TB-C) showed the efficacy of the self-attention mechanism in the AP task. However, the model that stands out the most is the WD-T aka *The Profiler*. The reason for this can be explained due to the *wide* branch of the architecture, which helps the correct profiling of users by contributing expressly engineered features for the AP problem in a manner that the *deep* element could not grasp. The novel WD-T architecture achieved top-quartile results in three AP shared tasks over English and Spanish datasets. Moreover, when compared only with DL strategies the WD-T outperformed them all.



True \ Predicted	female	male
female	1024	176
male	272	928

Acc: 0.8133
F1: 0.8130
P: 0.8154
R: 0.8133

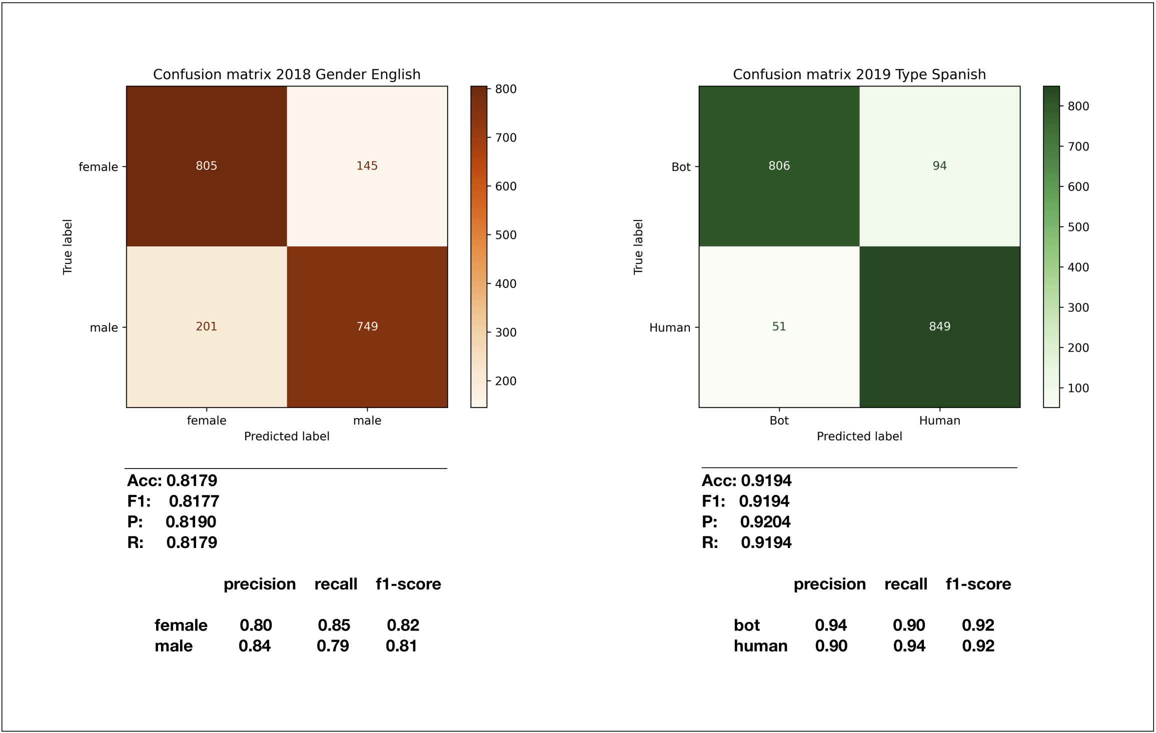
	precision	recall	f1-score
female	0.79	0.85	0.82
male	0.84	0.77	0.81

True \ Predicted	colombia	argentina	spain	venezuela	peru	chile	mexico
colombia	387	0	4	3	2	1	3
argentina	4	376	2	3	5	3	7
spain	2	1	373	11	5	2	6
venezuela	4	1	5	386	1	1	2
peru	5	2	7	9	370	3	4
chile	0	1	6	6	2	383	2
mexico	3	2	7	5	4	5	374

Acc: 0.9461
F1: 0.9461
P: 0.9466
R: 0.9461

	precision	recall	f1-score
argentina	0.96	0.97	0.96
chile	0.98	0.94	0.96
colombia	0.92	0.93	0.93
mexico	0.91	0.96	0.94
peru	0.95	0.93	0.94
spain	0.96	0.96	0.96
venezuela	0.94	0.94	0.94

Fig. 5.9. Confusion matrix for 2017 predictions tasks: English/Gender & Spanish/Lang. variety.



True \ Predicted	female	male
female	805	145
male	201	749

Acc: 0.8179
F1: 0.8177
P: 0.8190
R: 0.8179

	precision	recall	f1-score
female	0.80	0.85	0.82
male	0.84	0.79	0.81

True \ Predicted	Bot	Human
Bot	806	94
Human	51	849

Acc: 0.9194
F1: 0.9194
P: 0.9204
R: 0.9194

	precision	recall	f1-score
bot	0.94	0.90	0.92
human	0.90	0.94	0.92

Fig. 5.10. Confusion matrix for 2018: English/Gender & 2019: Spanish Type.

Chapter 6

General conclusions and further discussion

Digital text forensics handle the discovering of useful information in digital data banks. From the stand point of security, exposing Internet predatory activities or detecting fraud & cyber-terrorism are actions of great benefit. In addition, tracking suspicious activities over the Internet that could be potential threats to society is of great interest. However, there is a thin line that divides what might be considered a security concern and what can be viewed as an invasion of privacy. So developing proper technological solutions that are both effective and respectful of confidentiality is very important.

Among the several sub-categories of these forensic activities we can find *authorship analysis*, which includes *plagiarism detection*, *author identification* and *author profiling*. Particularly, *author profiling* is a computational task that attempts the prediction of personal characteristics of authors of digital texts (i.e., posts in blogs or social media) such as gender, age or even personality traits. In addition to the security concerns already mentioned, author profiling can also be helpful in improving customer service experience, diagnosis of neurological disorders (e.g., anxiety, depression) or detection of plagiarism, thus its relevance is evident. Author profiling is an ongoing natural language processing problem that is far from being resolved.

In the last decade author profiling tasks have been tackled with machine learning algorithms, using "traditional" techniques and in recent years employing deep learning approaches. In this thesis the author profiling problem was addressed from two fronts. First, proposing a novel and strong feature engineering methodology that could be categorized as classic machine learning. In such solution a new metric was devised (*rtv, relevance topic value*) for quantifying the importance of a word within a document. Then it was integrated along with other numerical statistics like *term frequency-inverse document frequency* to produce a weighting scheme of terms (encoded as word embeddings) using genetic programming, for constructing document embeddings (distributed representations of texts). The methodology attained competitive results in all tasks where it was tested, and even achieved State of the Art (SoA) results in some of them.

The second solution devised for the author profiling dilemma is based in deep learning methodologies. It fuses ideas like self-attention mechanisms from the novel *transformer* architecture and notions of blending *deep* and *linear* learning into one model, from *wide* & *deep* networks introduced circa 2016. Moreover, it provides a novel way to merge "stylistic" and "personal" information into the input word embeddings, which helps to increase the accuracy of the profiling. This novel approach called *The Profiler* aka the *Wide & Deep Transformer* (WD-T), achieved top-quartile results in three author profiling competitions, historically conquered by "traditional" (non-deep learning based) methodologies. In addition, when compared with only deep learning techniques, the WD-T obtained what can be considered SoA results, outperforming all strategies in those competitions and literature. Both approaches integrate proven useful features documented in the literature for the profiling of authors, in an original and effective way that performed adequately.

6.1 Quality analysis

The reader might question the reason to propose two different methodologies to address a unique problem. The reason for it lies in the potential practical applications of the scientific contributions of said approaches. When compared against each other, both strategies have advantage as well as disadvantages. Whilst the weighting-scheme strategy and the *rtv* metric introduced in **Chapter** 4 provides a novel yet "traditional" methodology that could be implemented with a moderate amount of computational resources. The second solution proposed in **Chapter** 5 is comprised of a more sophisticated architecture potentially capable of greater achievements but at a greater computational cost, which might be less of a problem in the years to come. Therefore, depending mainly in a potential application both strategies could be equally helpful.

But how do both approaches compare to each other in the profiling of authors when analyzing their *hits* and *misses*? It is noteworthy that some interesting findings were discovered. Take for instance the *gender* prediction task for the English language in the 2017 dataset. Both approaches tried more emphatically to learn words associated with topics typically connected with gender. For example in Fig.6.1 the hits and errors common to both approaches show a cross topic circumstance, where bigger concentrations of words like “time” and “trump” where associated with the prediction of *men* (c), and terms such as “thanks”, “day” and “love” where more correlated with the prediction of *women* (d). At the same time, the errors can be explained by the clouds (a) and (b), where it can be seen that samples with more quantity of terms associated with the opposite gender were misclassified.

On the other hand, in Fig. 6.2 the reader can appreciate the difference in performance of both: the *Wide & Deep Transformer* (WD-T) and the *Genetic Programming Evolved-Weighting Schemes* (GPE-WS). In the first row (word clouds (a) and (b)) we can observe that the GPE-WS gets right samples with more equally distributed terms for males, and highlights words that are not historically associated with one gender or the other for females, such as “persona”, “nunca” and “creo” (Spanish for “person”, “never” and “believe”). These word clouds also represent the samples where the WD-T was wrong. In addition, in the second row (word clouds (c) and (d)) it can be seen the WD-T pays more attention to words like “Dios”, “plan” and “gran” (Spanish for “God”, “plan” and “big”) for males, whilst also being more aware of terms like “trump”, “anyos” (“años” before pre-processing) and “pais” for females. It is interesting to note how the WD-T weighted terms commonly associated with gender the other way around, meaning that it would be more typical for a women to speak about God in Spanish speaking authors, while it would be commonplace for men to speak about “donald trump”. In this case the WD-T was able to perceive an open lexicon latent in the corpus, where nowadays in most western countries everybody can speak about anything, while at the same time the GPE-WS was not able to distinguish said words. Even though both methods were practically tied in accuracy performance for this dataset, it is noteworthy that the hits and misses of

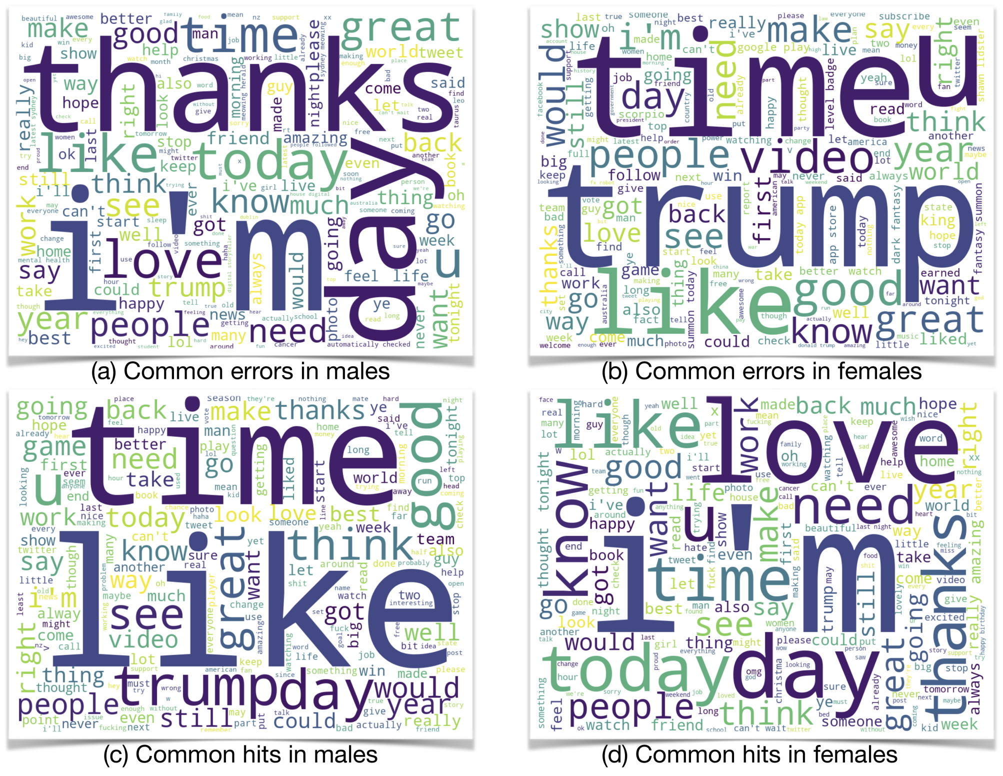
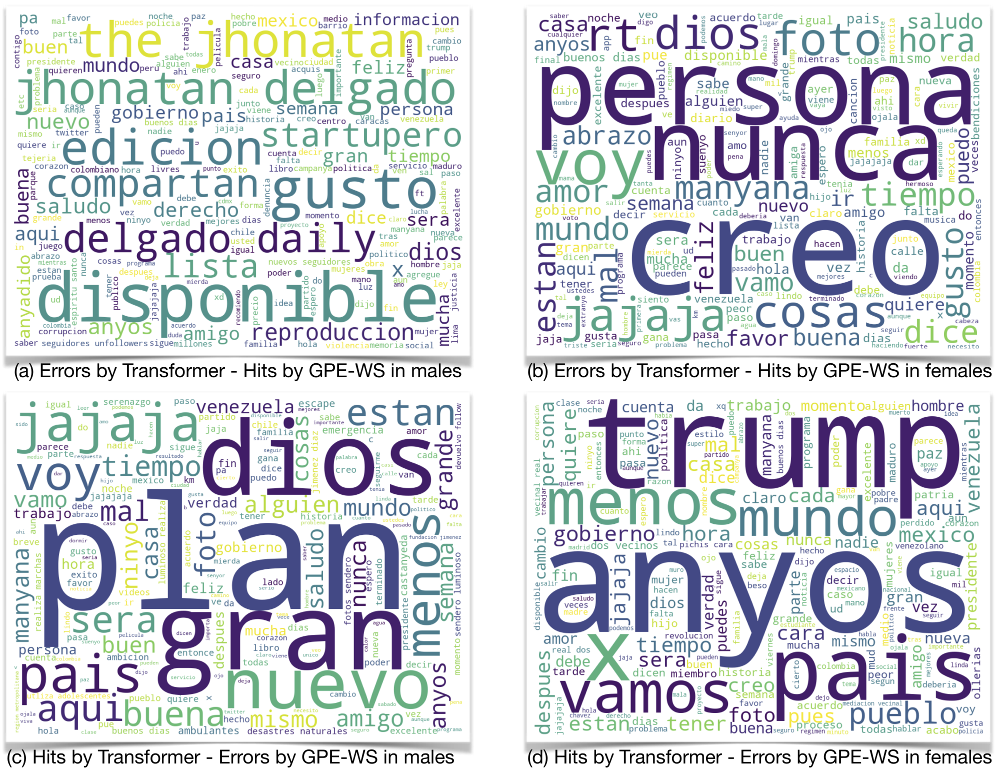


Fig. 6.1. Word clouds hits and errors for dataset 2017 gender English. Each word cloud represents the group of all samples for each category of error and hit.

each approach were very different.

For some NLP tasks (e.g., *depression* prediction) it has been documented that the length of texts might have some role to play [125]. For the author profiling task addressed in this thesis, the length of documents for both approaches in common hits and errors, as well as exclusive hits and misses was studied. Take for instance Fig. 6.3 and Fig. 6.4 for 2017 Spanish and 2018 English datasets respectively, we can recognize that the distribution of samples around the length is normal-like shaped. This fact gives the notion that for the author profiling task, at least in the experiments performed in this work, the length of a document it is not relevant, as long as the number of specimens available for training and testing is reasonable in amount.

Furthermore, in the 2019 author profiling task, only the second approach was tested. As mentioned in **Chapter** 5, two *transformer* based models were designed to



(a) Errors by Transformer - Hits by GPE-WS in males

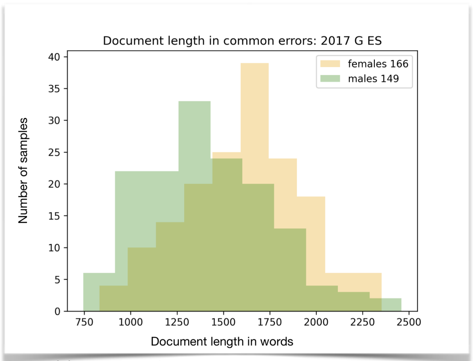
(b) Errors by Transformer - Hits by GPE-WS in females

(c) Hits by Transformer - Errors by GPE-WS in males

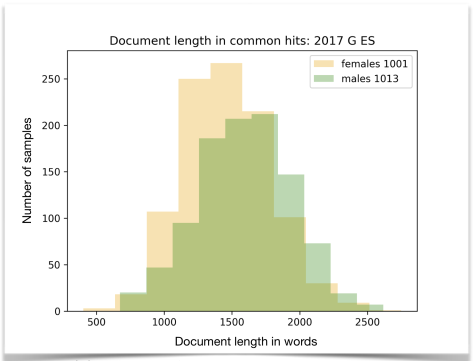
(d) Hits by Transformer - Errors by GPE-WS in females

Fig. 6.2. Word clouds hits and errors for dataset 2018 gender Spanish. Each word cloud represents the group of all samples for each category of error and hit.

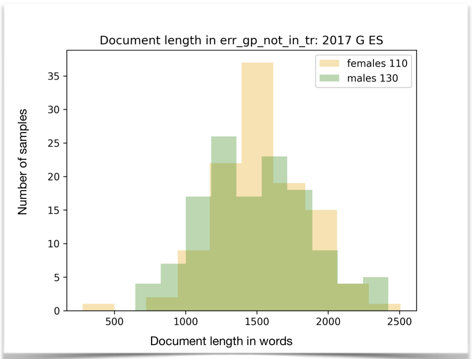
tackle author profiling: the WD-T as well as the *Transformer Based Contextualizer* (TB-C). In Fig. 6.5 we can observe the common hits and errors for both networks when distinguishing *bots* from *humans*. In the first row (word cloud (a)) it is evident that the error samples for humans included a high frequency of terms such as “ibm” and “seo” (*search engine optimization*), which might explain why both networks confused *humans* with *bots*. At the same time it can be seen in word cloud (b) that the word “mrw” (*my reaction when*) was confused by the networks as a term nowadays mostly associated with humans, thus producing errors when a *bot* employed it. In addition, in word cloud (c) we can not observe nothing extraordinary in the samples rightly classified as *humans*, in the mean time word cloud (d) depicts a more balanced frequency of terms, slightly standing out words like “software”, “engineer”, “project” and “manager”, which are technology terms likely correlated with *bots*.



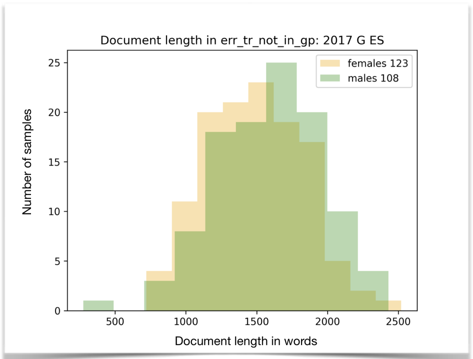
(a) Lenght of documents in common errors



(b) Lenght of documents in common hits



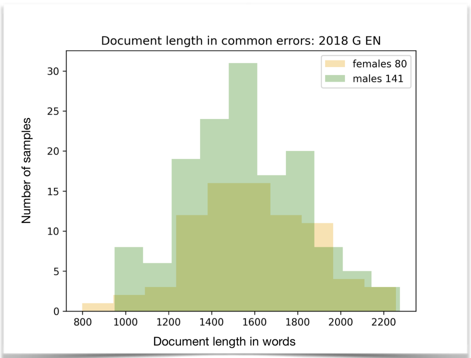
(c) Hits by Transformer - Errors by GPE-WS



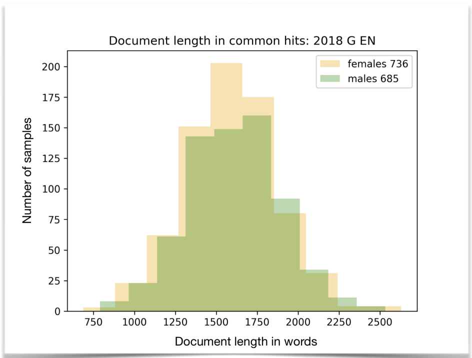
(d) Hits by GPE-WS - Errors by Transformer

Fig. 6.3. Document length for hit and error samples in the 2017 Spanish dataset for *gender* prediction.

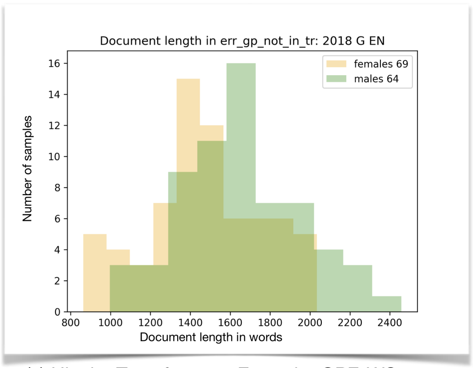
It was well documented in **Chapter** 5 the architecture and performance of both the WD-T and the TB-C. A natural question surfaced when comparing both networks, since the WD-T performed way better than the TB-C. What samples classified exclusively by either approach would look like? Fig. 6.6 answers this interrogation. We can see in word clouds (a) and (b) the hits exclusively made by the WD-T for *humans* and *bots* respectively. while in word clouds (c) and (d) the hits exclusively produced by the TB-C. Although we can debate whether or not some terms are more correlated with either *humans* or *bots* and what network got them better. What is really interesting is the fact that the WD-T "viewed" the samples for what they really represented: *humans* are expected to accentuate some words more than others (imprint a personal signature), whilst a *bot* will employ terms in a more "artificial" pace (i.e., a monotonous frequency). Hence we can hypothesized that the *wide* branch of



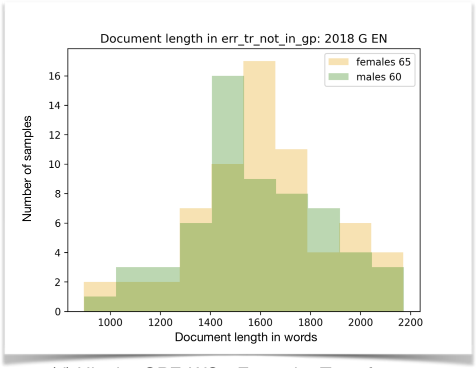
(a) Lenght of documents in common errors



(b) Lenght of documents in common hits



(c) Hits by Transformer - Errors by GPE-WS



(d) Hits by GPE-WS - Errors by Transformer

Fig. 6.4. Document length for hit and error samples in the 2018 English dataset for *gender* prediction.

the network can gather these nuances better than the *deep* side, complementing the overall performance.

Moreover, in Table 6.1 the reader can observe the mutual and reciprocal behavior of the approaches over some of the dataset collections. A metric called Maximum Possible Accuracy (MPA) was used to measure complementariness of the methodologies [126]. The MPA is computed by dividing the total amount of *hits* (successfully classified samples by at least on method) over the total number of *test* samples. Meaning that a hypothetical *perfect ensemble* of the techniques could deliver better results than each approach on its own. Also the common and exclusive *errors* are described. We can see in the first two rows (*gender* prediction for English and Spanish in 2017 and 2018 datasets) that the MPA is considerably higher than the best performer between the WD-T and the GPE-WS, thus suggesting they are somehow

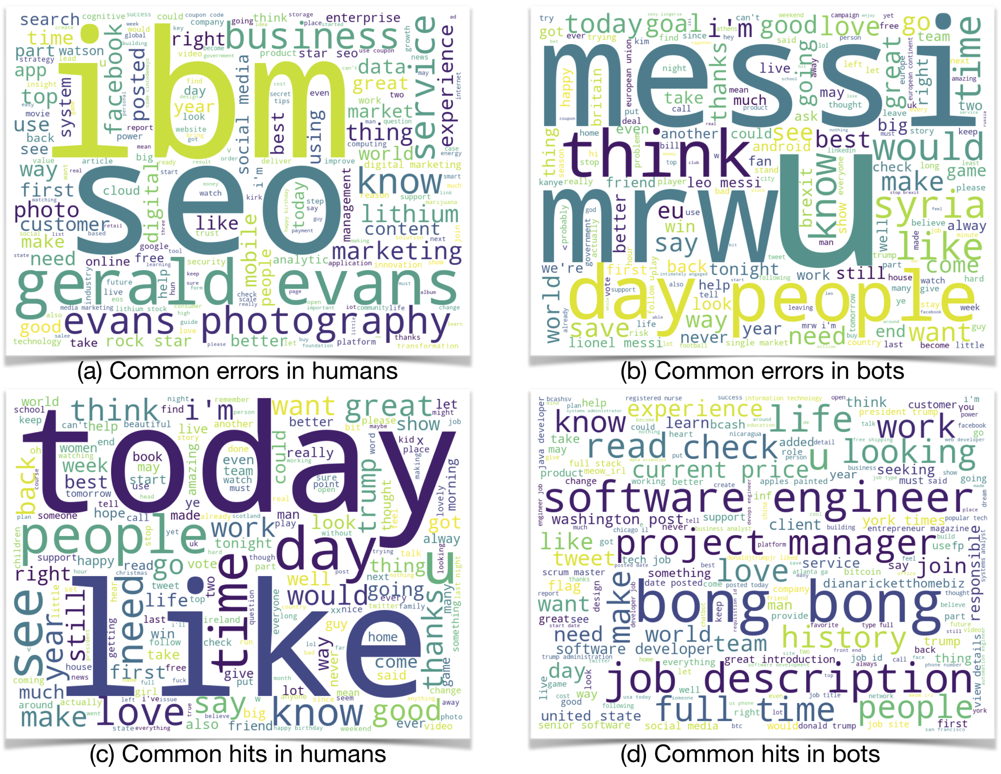
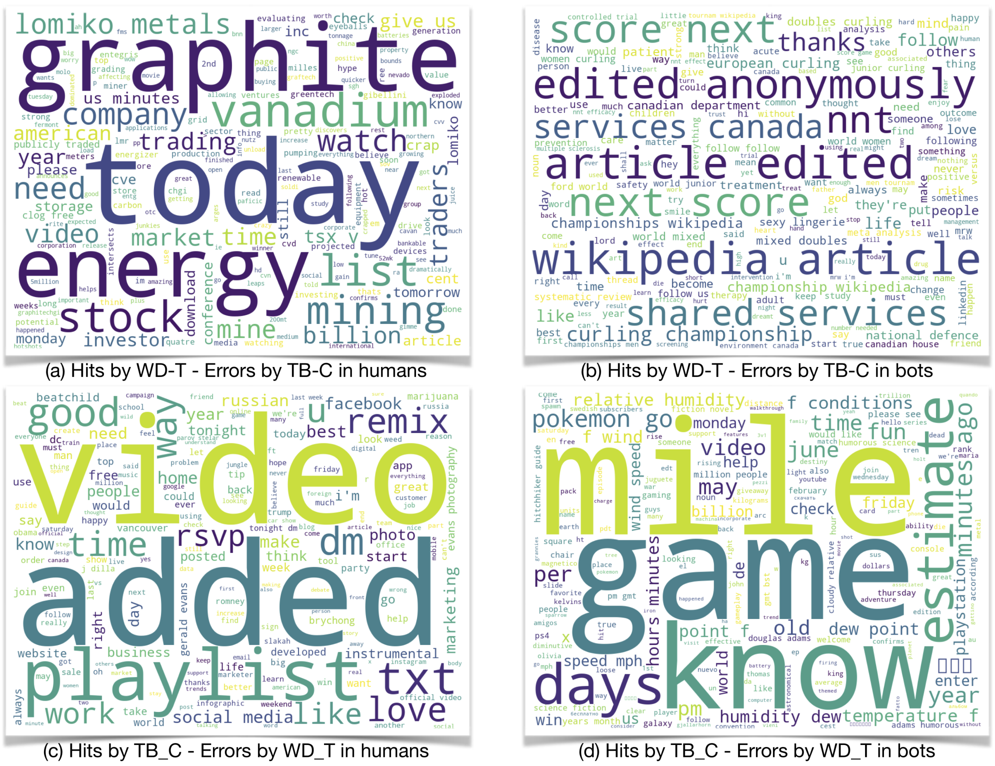


Fig. 6.5. Word clouds for samples in common hits and errors for the 2019 English dataset type prediction task (*bots* vs *humans*).

complementary. Also the common and exclusive misclassifications show a balanced distribution of *error* diversity, which is consistent with the MPA metric. Finally in last two rows we can see a comparison between both *transformer* based approaches (discussed in **Chapter** 5) for the English language. In this case we can see a less pronounced complementariness between the approaches, since the MPA is closer to the best performer classifier (WD-T). Also we can see that the mistakes exclusively made by the TB-C are greater in number (disbalanced), which suggests that the WD-T could be more effective on its own.



(a) Hits by WD-T - Errors by TB-C in humans (b) Hits by WD-T - Errors by TB-C in bots

(c) Hits by TB_C - Errors by WD_T in humans (d) Hits by TB_C - Errors by WD_T in bots

Fig. 6.6. Word clouds for samples in *transformer* exclusive hits and errors for the 2019 English dataset type prediction task (*bots* vs *humans*).

6.2 Future work

Even though both approaches provide competitive performance and results, it is recommended to further the research for the second approach. This methodology demonstrated more potential to address more varied NLP tasks, also it would be worth it to explore different and/or novel features to encode into the *wide* branch of the network, this would depend on the specific task addressed. Also, since the field of machine learning is continually evolving, and as of lately towards deep learning networks with strong attention mechanisms, the WD-T is very likely to be equally or even more successful in other NLP problems.

Table 6.1: Accuracy and error analysis over some representative dataset collections.

		English			*Spanish*		
Dataset	***Task/Classes***	***Accuracy***	***MPA***[a]	***Errors***[b]	***Accuracy***	***MPA***	***Errors***
2017	G[c]=2	-	0.8842	278	-	0.8875	315
WD-T[d]		0.8133	-	170	0.8050	-	231
GPE-WS[e]		0.8167	-	162	0.8018	-	240
2018	G=2	-	0.8837	221	-	0.8781	268
WD-T		0.8178	-	125	0.7918	-	190
GPE-WS		0.8174	-	133	0.7986	-	175
2019	T[f]=2	-	0.9352	171	-	-	-
WD-T		0.9163	-	26	-	-	-
TB-C[g]		0.8974	-	79	-	-	-
2019	G=3[h]	-	0.8458	407	-	-	-
WD-T		0.8044	-	71	-	-	-
TB-C		0.7781	-	221	-	-	-

[a]Maximum Possible Accuracy
[b]Common/Exclusive
[c]Gender: *female*, *male*
[d]Wide & Deep Transformer
[e]Genetic Programming Evolved-Weighting Scheme
[f]Type: *human* vs *bot*
[g]Transformer Based Contextualizer
[h]*female*, *male*, *bot*

6.3 Final discussion

What could be the philosophical implications of the proposed methodologies? The answer might be found in the problem addressed originally. Why is author profiling important nowadays? Is it really that relevant? Recently the discussion about ethics in the Machine Learning (ML) field has become controversial. Much like several current topics, ML it is not exempt of debate. With the current state of affairs regarding subjects such as diversity among the workplace or the equality between gender, there has been a discussion whether ML algorithms are biased towards historically privileged segments of the population. Several technology companies like Google, Facebook, Twitter or Apple have included a department of ethics within their artificial intelligence department. In addition, language models based in the *transformer* architecture such as BERT have been criticized. It has even been said that these language models are nothing short of "glorified parrots", since they are very efficiently trained to mimic the human language with impressive results. What is interesting about these arguments is that crowds tend to behave similarly when behind a social media account. More often than not, persons tend to hide their true opinions in social media outlets fearing they could be misunderstood or even “cancelled”. This begets the question of whether language models are not simply a magnifying mirror in which we can see our true reflection as society.

About the bias in ML algorithms, it is true that prediction models tend to behave with prejudice sometimes. Take for instance the case of a recurrently misclassified sample by both the WD-T and the TB-C. A *woman* in the 2017 dataset was constantly wrongly categorized as a *man*. When her posts were analyzed more deeply it was evident that the woman was a Canadian professional women’s soccer player. She frequently posted about local and national women’s soccer teams. Neither model (e.g., WD-T, TB-C) was able to properly identify that the woman was positively talking about themes that as of now are not necessarily associated with men. We can argue that models could be learning with certain bias, but certainly they can not be accused of sexism. Currently Deep Learning networks are the SoA ML algorithms in many application areas of society. They are great learners, but just like with a small child, we must be careful with what we teach them, we could produce negatively prejudiced predictors or incredible helpful assistants.

As already stated, author profiling remains an ongoing and relevant natural language processing problem. The research to find better and more efficient solutions continues to grow. In this thesis, I believe the reader could find two novel and sound proposals to successfully approach the very interesting author profiling conundrum. Nonetheless, beyond the methodologies themselves, the author profiling problem could also evolve. It would be very interesting to investigate the profiling of certain segments of society. Instead of forecasting characteristics of individuals, the *Profiler* network, or any ML algorithm for that matter, could predict particular features of digital communities. The potential applications would be of great interest. Imagine the possibility to anticipate violent uprisings, racist movements or any kind of dangerous threat. Although that would be useful, one have to mind the potential ethic implications.

References

[1] M. Potthast, P. Rosso, E. Stamatatos, and B. Stein, "A decade of shared tasks in digital text forensics at pan," in *Advances in Information Retrieval* (L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, and D. Hiemstra, eds.), (Cham), pp. 291–300, Springer International Publishing, 2019.

[2] S. M. Rezaeinia, R. Rahmani, A. Ghodsi, and H. Veisi, "Sentiment analysis based on improved pre-trained word embeddings," *Expert Systems with Applications*, vol. 117, pp. 139 – 147, 2019.

[3] M. Fatima, K. Hasan, S. Anwar, and R. M. A. Nawab, "Multilingual author profiling on facebook," *Information Processing & Management*, vol. 53, no. 4, pp. 886 – 904, 2017.

[4] S. Sengupta, S. Basak, P. Saikia, S. Paul, V. Tsalavoutis, F. Atiah, V. Ravi, and A. Peters, "A review of deep learning with special emphasis on architectures, applications and recent trends," *Knowledge-Based Systems*, vol. 194, p. 105596, 2020.

[5] F. Rangel and P. Rosso, "On the impact of emotions on author profiling," *Information Processing & Management*, vol. 52, no. 1, pp. 73–92, 2016. cited By 41.

[6] M. E. Aragón, A. P. López-Monroy, L. C. González-Gurrola, and M. Montes-y Gómez, "Detecting depression in social media using fine-grained emotions," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 1481–1486, Association for Computational Linguistics, June 2019.

[7] M. E. Aragón, A. P. López-Monroy, L. C. González, and M. Montes-y Gómez, "Attention to emotions: Detecting mental disorders in social media," in *Text, Speech, and Dialogue* (P. Sojka, I. Kopeček, K. Pala, and A. Horák, eds.), (Cham), pp. 231–239, Springer International Publishing, 2020.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 3111–3119, Curran Associates, Inc., 2013.

[9] M. Kocher and J. Savoy, "Distance measures in author profiling," *Information Processing & Management*, vol. 53, no. 5, pp. 1103 – 1119, 2017.

[10] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pp. II–1188–II–1196, JMLR.org, 2014.

[11] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 957–966, JMLR.org, 2015.

[12] R. Bruns, J. Dunkel, and N. Offel, "Learning of complex event processing rules with genetic programming," *Expert Systems with Applications*, vol. 129, pp. 186–199, 2019.

[13] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide deep learning for recommender systems," 2016.

[14] B. Stein, M. Koppel, and E. Stamatatos, "Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection (PAN-07)," in *Proceedings of the SIGIR'07 International Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection* (B. Stein, M. Koppel, and E. Stamatatos, eds.), Proceedings of the SIGIR'07 International Workshop, CEUR-WS.org, July 2007.

[15] F. Rangel, P. Rosso, M. Montes-y-Gómez, M. Potthast, and B. Stein, "Overview of the 6th Author Profiling Task at PAN 2018: Multimodal Gender Identification in Twitter," in *Working Notes Papers of the CLEF 2018 Evaluation Labs* (L. Cappellato, N. Ferro, J.-Y. Nie, and L. Soulier, eds.), CEUR Workshop Proceedings, CLEF and CEUR-WS.org, Sept. 2018.

[16] F. Rangel, F. C. P. Rosso, M. Potthast, B. Stein, and W. Daelemans, "Daelemans, w.: Overview of the 3rd author profiling task at pan 2015," in *CLEF 2015 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings, CEUR-WS.org (Sep 2015)*, 2015.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* The MIT Press, 2016. `http://www.deeplearningbook.org`.

[18] E. Stamatatos, F. Rangel, M. Tschuggnall, M. Kestemont, P. Rosso, B. Stein, and M. Potthast, "Overview of PAN-2018: Author Identification, Author Profiling, and Author Obfuscation," in *Experimental IR Meets Multilinguality, Multimodality, and Interaction. 9th International Conference of the CLEF Initiative (CLEF 18)* (P. Bellot, C. Trabelsi, J. Mothe, F. Murtagh, J. Nie, L. Soulier, E. Sanjuan, L. Cappellato, and N. Ferro, eds.), (Berlin Heidelberg New York), Springer, Sept. 2018.

[19] G. K. Mikros and K. Perifanos, "Gender identification in modern greek tweets," in *Recent Contributions to Quantitative Linguistics*, pp. 75–88, 2015.

[20] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of CNN and RNN for natural language processing," *CoRR*, vol. abs/1702.01923, 2017.

[21] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. of NAACL*, 2018.

[22] T. Dubovičienė and P. Skorupa, "The analysis of some stylistic features of english advertising slogans," *Žmogus ir žodis*, vol. 16, no. 3, pp. 61–75, 2014.

[23] M. Lai, A. T. Cignarella, D. I. H. Farías], C. Bosco, V. Patti, and P. Rosso, "Multilingual stance detection in social media political debates," *Computer Speech Language*, vol. 63, p. 101075, 2020.

[24] X. Li, P. Wu, and W. Wang, "Incorporating stock prices and news sentiments for stock market prediction: A case of hong kong," *Information Processing Management*, p. 102212, 2020.

[25] P. Meel and D. K. Vishwakarma, "Fake news, rumor, information pollution in social media and web: A contemporary survey of state-of-the-arts, challenges and opportunities," *Expert Systems with Applications*, vol. 153, p. 112986, 2020.

[26] L. Wu, Y. Rao, A. Nazir, and H. Jin, "Discovering differential features: Adversarial learning for information credibility evaluation," *Information Sciences*, vol. 516, pp. 453 – 473, 2020.

[27] C. Li, J. Bai, Z. Wenjun, and Y. Xihao, "Community detection using hierarchical clustering based on edge-weighted similarity in cloud environment," *Information Processing & Management*, vol. 56, no. 1, pp. 91 – 109, 2019.

[28] N. Sarma, S. R. Singh, and D. Goswami, "Influence of social conversational features on language identification in highly multilingual online conversations," *Information Processing & Management*, vol. 56, no. 1, pp. 151 – 166, 2019.

[29] P. Sánchez and A. Bellogín, "Building user profiles based on sequences for content and collaborative filtering," *Information Processing & Management*, vol. 56, no. 1, pp. 192 – 211, 2019.

[30] A. Khatua, A. Khatua, and E. Cambria, "A tale of two epidemics: Contextual word2vec for classifying twitter streams during outbreaks," *Information Processing & Management*, vol. 56, no. 1, pp. 247 – 257, 2019.

[31] A. Capatina, M. Kachour, J. Lichy, A. Micu, A.-E. Micu, and F. Codignola, "Matching the future capabilities of an artificial intelligence-based software for social media marketing with potential users' expectations," *Technological Forecasting and Social Change*, vol. 151, p. 119794, 2020.

[32] F. Rangel, P. Rosso, M. Koppel, E. Stamatatos, and G. Inches, "Overview of the author profiling task at pan 2013," in *CLEF 2013 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings, CEUR-WS.org (Sep 2013)*, 2013.

[33] M. Á. Á. Carmona, A. P. López-Monroy, M. M. y Gómez, L. V. Pineda, and I. V. Meza, "Evaluating topic-based representations for author profiling in social media," in *IBERAMIA*, 2016.

[34] J. Calle-Martín and A. Miranda-García, "Stylometry and authorship attribution: Introduction to the special issue," *English Studies*, vol. 93, no. 3, pp. 251–258, 2012.

[35] E. Stamatatos, "A survey of modern authorship attribution methods," *J. Am. Soc. Inf. Sci. Technol.*, vol. 60, pp. 538–556, Mar. 2009.

[36] O. Fourkioti, S. Symeonidis, and A. Arampatzis, "Language models and fusion for authorship attribution," *Information Processing Management*, vol. 56, no. 6, p. 102061, 2019.

[37] D. Adair, *Fame and the founding fathers : essays*. Indianapolis : Liberty Fund, 1998. Originally published: New York : Norton for the Institute of Early American History and Culture at Williamsburg, 1974.

[38] C. De Boom, S. Van Canneyt, T. Demeester, and B. Dhoedt, "Representation learning for very short texts using weighted word embedding aggregation," *Pattern Recogn. Lett.*, vol. 80, pp. 150–156, Sept. 2016.

[39] H. Saif, Y. He, M. Fernandez, and H. Alani, "Contextual semantics for sentiment analysis of twitter," *Information Processing & Management*, vol. 52, no. 1, pp. 5–19, 2016. cited By 123.

[40] E. Fersini, E. Messina, and F. Pozzi, "Expressive signals in social media languages to improve polarity detection," *Information Processing & Management*, vol. 52, no. 1, pp. 20–35, 2016. cited By 23.

[41] R. Ortega-Mendoza, A. López-Monroy, A. Franco-Arcega, and M. Montes, "Emphasizing personal information for author profiling: New approaches for term selection and weighting," *Knowledge-Based Systems*, vol. 145, pp. 169 – 181, 04 2018.

[42] A. P. López-Monroy, M. Montes-y-Gómez, H. J. Escalante, L. V. Pineda, and E. Villatoro-Tello, "Inaoe's participation at pan'13: Author profiling task notebook for PAN at CLEF 2013," in *Working Notes for CLEF 2013 Conference , Valencia, Spain, September 23-26, 2013.*, 2013.

[43] A. P. López-Monroy, M. MontesY-Gómez, H. J. Escalante, and L. Villaseñor Pineda, "Using intra-profile information for author profiling," in *Working Notes of CLEF 2014 - Conference and Labs of the Evaluation Forum*, vol. 1180, 09 2014.

[44] M. A. Álvarez-Carmona, A. P. López-Monroy, M. MontesY-Gómez, L. Villaseñor Pineda, and H. J. Escalante, "Inaoe's participation at pan'15: Author profiling task.," in *CLEF (Working Notes)* (L. Cappellato, N. Ferro, G. J. F. Jones, and E. SanJuan, eds.), vol. 1391 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2015.

[45] M. B. O. Vollenbroek, T. Carlotto, T. Kreutz, M. Medvedeva, C. Pool, J. Bjerva, H. Haagsma, and M. Nissim, "Gronup: Groningen user profiling," in *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016.*, pp. 846–857, 2016.

[46] A. Basile, G. Dwyer, M. Medvedeva, J. Rawee, H. Haagsma, and M. Nissim, "N-gram: New groningen author-profiling model," *CoRR*, vol. abs/1707.03764, 2017.

[47] S. Daneshvar and D. Inkpen, "Gender identification in twitter using n-grams and lsa: Notebook for pan at clef 2018," in *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018.*, 2018.

[48] J. Pizarro, "Using n-grams to detect bots on twitter," in *CLEF*, 2019.

[49] P. Rosso, M. Potthast, B. Stein, E. Stamatatos, F. Rangel, and W. Daelemans, *Evolution of the PAN Lab on Digital Text Forensics*, pp. 461–485. Cham: Springer International Publishing, 2019.

[50] Y. Miura, T. Taniguchi, M. Taniguchi, and T. Ohkuma, "Author profiling with word+character neural attention network," in *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017* (L. Cappellato, N. Ferro, L. Goeuriot, and T. Mandl, eds.), vol. 1866 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017.

[51] M. Hosseinia and A. Mukherjee, "A parallel hierarchical attention network for style change detection: Notebook for PAN at CLEF 2018," in *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018* (L. Cappellato, N. Ferro, J. Nie, and L. Soulier, eds.), vol. 2125 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018.

[52] M. Polignano, M. G. de Pinto, P. Lops, and G. Semeraro, "Identification of bot accounts in twitter using 2d cnns on user-generated contents," in *CLEF*, 2019.

[53] U. Kamath, J. Liu, and J. Whitaker, *Deep Learning for NLP and Speech Recognition*. Springer Publishing Company, Incorporated, 1st ed., 2019.

[54] G. Rossiello, P. Basile, and G. Semeraro, "Centroid-based text summarization through compositionality of word embeddings," in *Proceedings of the MultiLing 2017 Workshop on Summarization and Summary Evaluation Across Source Types and Genres*, (Valencia, Spain), pp. 12–21, Association for Computational Linguistics, Apr. 2017.

[55] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[56] P. Michel, A. Ravichander, and S. Rijhwani, "Does the geometry of word embeddings help document classification? a case study on persistent homology-based representations," in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, (Vancouver, Canada), pp. 235–240, Association for Computational Linguistics, Aug. 2017.

[57] M. Chen, "Efficient vector representation for documents through corruption," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[58] L. Wu, I. E. Yen, K. Xu, F. Xu, A. Balakrishnan, P. Chen, P. Ravikumar, and M. J. Witbrock, "Word mover's embedding: From word2vec to document embedding," *CoRR*, vol. abs/1811.01713, 2018.

[59] H. J. Escalante, M. A. García-Limón, A. Morales-Reyes, M. Graff, M. Montes-y Gómez, E. F. Morales, and J. Martínez-Carranza, "Term-weighting learning via genetic programming for text classification," *Know.-Based Syst.*, vol. 83, pp. 176–189, July 2015.

[60] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]," *IEEE Computational Intelligence Magazine*, vol. 13, pp. 55–75, 08 2018.

[61] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 6294–6305, Curran Associates, Inc., 2017.

[62] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 24, pp. 694–707, Apr. 2016.

[63] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Towards universal paraphrastic sentence embeddings," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[64] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

[65] S. Marsland, *Machine Learning: An Algorithmic Perspective, Second Edition.* Chapman & Hall/CRC, 2nd ed., 2014.

[66] S. Liu, K. Lee, and I. Lee, "Document-level multi-topic sentiment classification of email data with bilstm and data augmentation," *Knowledge-Based Systems*, vol. 197, p. 105918, 2020.

[67] V. Chow, "Predicting auction price of vehicle license plate with deep recurrent neural network," *Expert Systems with Applications*, vol. 142, p. 113008, 2020.

[68] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 5998–6008, Curran Associates, Inc., 2017.

[69] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media, 2019.

[70] P. Liu, L. Zhang, and J. A. Gulla, "Dynamic attention-based explainable recommendation with textual and visual fusion," *Information Processing Management*, p. 102099, 2019.

[71] Y. Li, T. Liu, J. Hu, and J. Jiang, "Topical co-attention networks for hashtag recommendation on microblogs," *Neurocomputing*, vol. 331, pp. 356 – 365, 2019.

[72] X. Luo and Z. Chen, "English text quality analysis based on recurrent neural network and semantic segmentation," *Future Generation Computer Systems*, 2020.

[73] Y. Liang, T. Qian, and H. Yu, "Artan: Align reviews with topics in attention network for rating prediction," *Neurocomputing*, vol. 403, pp. 337 – 347, 2020.

[74] G. Rizzo and T. H. M. Van, "Adversarial text generation with context adapted global knowledge and a self-attentive discriminator," *Information Processing Management*, p. 102217, 2020.

[75] J. Ángel González, L.-F. Hurtado, and F. Pla, "Transformer based contextualization of pre-trained word embeddings for irony detection in twitter," *Information Processing Management*, vol. 57, no. 4, p. 102262, 2020.

[76] J. Worsham and J. Kalita, "Multi-task learning for natural language processing in the 2020s: where are we going?," *Pattern Recognition Letters*, 2020.

[77] M. S. Akhtar, T. Garg, and A. Ekbal, "Multi-task learning for aspect term extraction and aspect sentiment classification," *Neurocomputing*, vol. 398, pp. 247 – 256, 2020.

[78] G. Lu, X. Zhao, J. Yin, W. Yang, and B. Li, "Multi-task learning using variational auto-encoder for sentiment classification," *Pattern Recognition Letters*, vol. 132, pp. 115 – 122, 2020. Multiple-Task Learning for Big Data (MTL4BD).

[79] H. Luo, J. Cai, K. Zhang, R. Xie, and L. Zheng, "A multi-task deep learning model for short-term taxi demand forecasting considering spatiotemporal dependences," *Journal of Traffic and Transportation Engineering (English Edition)*, 2020.

[80] T. Nguyen, M. Raghu, and S. Kornblith, "Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth," 2020.

[81] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

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

[83] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[84] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *In EMNLP*, 2014.

[85] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[86] R. López-Santillán, M. Montes-Y-Gómez, L. C. González-Gurrola, G. Ramírez-Alonso, and O. Prieto-Ordaz, "Richer document embeddings for author profiling tasks based on a heuristic search," *Information Processing Management*, vol. 57, no. 4, p. 102227, 2020.

[87] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," *ArXiv*, vol. abs/1910.03771, 2019.

[88] M. E. Peters, S. Ruder, and N. A. Smith, "To tune or not to tune? adapting pretrained representations to diverse tasks," *CoRR*, vol. abs/1903.05987, 2019.

[89] J. R. Koza, "Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems," tech. rep., Stanford, CA, USA, 1990.

[90] T. Stephens, "gplearn documentation," pp. 1–55, apr 2019.

[91] F. Rangel, P. Rosso, B. Verhoeven, W. Daelemans, M. Pottast, and B. Stein, "Overview of the 4th author profiling task at pan 2016: Cross-genre evaluations," in *CLEF 2016 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings. CEUR-WS.org* (B. K., C. L., F. N., and M. C, eds.), 2016.

[92] F. Rangel, P. Rosso, M. Potthast, and B. Stein, "Overview of the 5th author profiling task at pan 2017: Gender and language variety identification in twitter," in *CLEF 2017 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings. CEUR-WS.org* (C. L., F. N., G. L, and M. T, eds.), 2017.

[93] I. Pervaz, I. Ameer, A. Sittar, and R. M. A. Nawab, "Identification of author personality traits using stylistic features: Notebook for pan at clef 2015.," in *CLEF (Working Notes)* (L. Cappellato, N. Ferro, G. J. F. Jones, and E. San-Juan, eds.), vol. 1391 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2015.

[94] Y. Kiprov, M. Hardalov, P. Nakov, and I. Koychev, "Su@pan'2015: Experiments in author profiling.," in *CLEF (Working Notes)* (L. Cappellato, N. Ferro, G. J. F. Jones, and E. SanJuan, eds.), vol. 1391 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2015.

[95] S. Ashraf, H. R. Iqbal, and R. M. A. Nawab, "Cross-genre author profile prediction using stylometry-based approach," in *CLEF 2016 Working Notes. CEUR Workshop Proceedings (CEUR-WS.org)* (K. Balog, L. Cappellato, N. Ferro, and C. Macdonald, eds.), http://ceur-ws.org/Vol-1609/, 2016.

[96] K. Bougiatiotis and A. Krithara, "Author profiling using complementary second order attributes and stylometric features," in *CLEF 2016 Working Notes. CEUR Workshop Proceedings (CEUR-WS.org)* (K. Balog, L. Cappellato, N. Ferro, and C. Macdonald, eds.), http://ceur-ws.org/Vol-1609/, 2016.

[97] A. P. Lopez-Monroy, M. Montes-Y-Gomez, H. J. Escalante, L. Villasenor-Pineda, and E. Villatoro-Tello, "Inaoe's participation at pan'13: author profiling task—notebook for pan at clef 2013," in *CLEF 2013 Evaluation Labs and Workshop – Working Notes Papers* (P. Forner, R. Navigli, and D. Tufis, eds.), 2013.

[98] R. B. Deyab, J. Duarte, and T. Gonçalves, "Author profiling using support vector machines," in *CLEF 2016 Working Notes. CEUR Workshop Proceedings (CEUR-WS.org)* (K. Balog, L. Cappellato, N. Ferro, and C. Macdonald, eds.), http://ceur-ws.org/Vol-1609/, 2016.

[99] M. Agrawal and T. Gonçalves, "Age and gender identification using stacking for classification," in *CLEF 2016 Working Notes. CEUR Workshop Proceedings (CEUR-WS.org)* (K. Balog, L. Cappellato, N. Ferro, and C. Macdonald, eds.), http://ceur-ws.org/Vol-1609/, 2016.

[100] R. K. Bayot and T. Gonçalves, "Author profiling using svms and word embedding averages," in *CLEF 2016 Working Notes. CEUR Workshop Proceedings (CEUR-WS.org)* (K. Balog, L. Cappellato, N. Ferro, and C. Macdonald, eds.), http://ceur-ws.org/Vol-1609/, 2016.

[101] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010. `http://is.muni.cz/publication/884893/en`.

[102] F. Rangel, P. Rosso, I. Chugur, M. Potthast, M. Trenkmann, B. Stein, B. Verhoeven, and W. Daelemans, "Overview of the 2nd author profiling task at pan 2014," in *CLEF 2014 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings, CEUR-WS.org (Sep 2014)*, 2014.

[103] L. R. Goldberg, "The development of markers for the big-five factor structure.," *Psychological Assessment*, vol. 4, no. 1, pp. 26–42, 1992.

[104] J. Pennebaker, *The Secret Life of Pronouns: What Our Words Say About Us.* Bloomsbury USA, 2011.

[105] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," 2015.

[106] M. Franco-Salvador, N. Plotnikova, N. Pawar, and Y. Benajiba, "Subword-based deep averaging networks for author profiling in social media.," in *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017* (L. Cappellato, N. Ferro, L. Goeuriot, and T. Mandl, eds.), vol. 1866 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017.

[107] D. Kodiyan, F. Hardegger, S. Neuhaus, and M. Cieliebak, "Author profiling with bidirectional rnns using attention with grus," in *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017* (L. Cappellato, N. Ferro, L. Goeuriot, and T. Mandl, eds.), vol. 1866 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017.

[108] N. Schaetti, "Unine at CLEF 2017: TF-IDF and deep-learning for author profiling," in *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017* (L. Cappellato, N. Ferro, L. Goeuriot, and T. Mandl, eds.), vol. 1866 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017.

[109] S. Sierra, M. Montes-y-Gómez, T. Solorio, and F. A. González, “Convolutional neural networks for author profiling in PAN 2017,” in *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017* (L. Cappellato, N. Ferro, L. Goeuriot, and T. Mandl, eds.), vol. 1866 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017.

[110] R. K. Bayot and T. Gonçalves, “Multilingual author profiling using lstms: Notebook for PAN at CLEF 2018,” in *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018* (L. Cappellato, N. Ferro, J. Nie, and L. Soulier, eds.), vol. 2125 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018.

[111] M. Martinc, B. Skrlj, and S. Pollak, “Multilingual gender classification with multi-view deep learning: Notebook for PAN at CLEF 2018,” in *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018* (L. Cappellato, N. Ferro, J. Nie, and L. Soulier, eds.), vol. 2125 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018.

[112] N. Schaetti, “Character-based convolutional neural network and resnet18 for twitter author profiling: Notebook for PAN at CLEF 2018,” in *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018* (L. Cappellato, N. Ferro, J. Nie, and L. Soulier, eds.), vol. 2125 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018.

[113] E. Sezerer, O. Polatbilek, Ö. Sevgili, and S. Tekir, “Gender prediction from tweets with convolutional neural networks: Notebook for PAN at CLEF 2018,” in *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018* (L. Cappellato, N. Ferro, J. Nie, and L. Soulier, eds.), vol. 2125 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018.

[114] O. Halvani and P. Marquardt, “An unsophisticated neural bots and gender profiling system,” in *Working Notes of CLEF 2019 - Conference and Labs of the Evaluation Forum, Lugano, Switzerland, September 9-12, 2019* (L. Cappellato, N. Ferro, D. E. Losada, and H. Müller, eds.), vol. 2380 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019.

[115] J. Petrík and D. Chuda, “Bots and gender profiling with convolutional hierarchical recurrent neural network notebook for pan at clef 2019,” 10 2019.

[116] F. Bolonyai, J. Buda, and E. Katona, “Bot or not: A two-level approach in author profiling notebook for pan at clef 2019,” 01 2019.

[117] C. Onose, C. Nedelcu, D. Cercel, and S. Trausan-Matu, "A hierarchical attention network for bots and gender profiling," in *Working Notes of CLEF 2019 - Conference and Labs of the Evaluation Forum, Lugano, Switzerland, September 9-12, 2019* (L. Cappellato, N. Ferro, D. E. Losada, and H. Müller, eds.), vol. 2380 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019.

[118] R. F. S. Dias and I. Paraboni, "Combined cnn+rnn bot and gender profiling," in *CLEF*, 2019.

[119] M. Färber, A. Qurdina, and L. Ahmedi, "Identifying twitter bots using a convolutional neural network," in *CLEF*, 2019.

[120] E. Loper and S. Bird, "Nltk: The natural language toolkit," *CoRR*, vol. cs.CL/0205028, 2002.

[121] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A Python natural language processing toolkit for many human languages," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020.

[122] F. M. R. Pardo and P. Rosso, "Overview of the 7th author profiling task at pan 2019: Bots and gender profiling in twitter," in *CLEF*, 2019.

[123] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," 2019.

[124] N. Kitaev, Łukasz Kaiser, and A. Levskaya, "Reformer: The efficient transformer," 2020.

[125] J. d. J. Titla Tlatelpa, *Detección de depresión en redes sociales considerando información del perfil de los usuarios.* PhD thesis, INAOE, 2020.

[126] M. E. Aragón, H. J. Jarquín-Vásquez, M. Montes-y-Gómez, H. J. Escalante, L. V. Pineda, H. Gómez-Adorno, J. P. Posadas-Durán, and G. Bel-Enguix, "Overview of MEX-A3T at iberlef 2020: Fake news and aggressiveness analysis in mexican spanish," in *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2020) co-located with 36th Conference of the Spanish Society for Natural Language Processing (SEPLN 2020), Málaga, Spain, September 23th, 2020* (M. Á. G. Cumbreras, J. Gonzalo, E. M. Cámara, R. Martínez-Unanue, P. Rosso, S. M. J. Zafra, J. A. O. Zambrano, A. Miranda, J. P. Zamorano, Y. Gutiérrez, A. Rosá, M. Montes-y-Gómez, and M. G. Vega, eds.), vol. 2664 of *CEUR Workshop Proceedings*, pp. 222–235, CEUR-WS.org, 2020.

[127] M. Casavantes, R. López, and L. C. González, "Uach at mex-a3t 2019: Preliminary results on detecting aggressive tweets by adding author information via an unsupervised strategy," Sept. 2019.

[128] M. Casavantes and R. López, "Uach-inaoe at hasoc 2019: detecting aggressive tweets by incorporating authors' traits as descriptors," in *FIRE 2019 Working Notes*, CEUR-WS.org, Dec. 2019.

[129] R. López, L. C. González Gurrola, L. Trujillo, O. Prieto, G. Ramírez, A. Posada, P. Juárez-Smith, and L. Méndez, "How am i driving? using genetic programming to generate scoring functions for urban driving behavior," *Mathematical and Computational Applications*, vol. 23, no. 2, 2018.

[130] J. R. López, L. C. González, J. Wahlström, M. M. y Gómez, L. Trujillo, and G. Ramírez-Alonso, "A genetic programming approach for driving score calculation in the context of intelligent transportation systems," *IEEE Sensors Journal*, vol. 18, pp. 7183–7192, Sept 2018.

[131] O. Prieto-Ordáz, G. Ramírez-Alonso, L. C. González, R. López-Santillán, and M. Montes-y-Gómez, "Brain tumor segmentation using an encoder-decoder network with a multiscale feature module," in *2020 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, vol. 4, pp. 1–6, 2020.

[132] J. Aguilera, L. C. González, M. M. y Gómez, R. López, and H. J. Escalante, "From neighbors to strengths - the k-strongest strengths (kss) classification algorithm," *Pattern Recognition Letters*, vol. 136, pp. 301 – 308, 2020.

Appendices

Appendix A

List of publications

A.1 Thesis Related

A.1.1 Journals

- **Roberto López-Santillán**, Manuel Montes-Y-Gómez, Luis Carlos González-Gurrola, Graciela Ramírez-Alonso, Olanda Prieto-Ordaz, ***Richer Document Embeddings for Author Profiling tasks based on a heuristic search***, Elsevier *Information Processing & Management* Journal. [86]. Indexed by the Journal Citation Reports (Clarivate Analytics, 2019), Impact Factor: 4.787, Q1, 2020, 102227, ISSN 0306-4573,
https://doi.org/10.1016/j.ipm.2020.102227.
(http://www.sciencedirect.com/science/article/pii/S0306457319306466)

A.1.2 Congress

- **López-Santillán, R.**, Gonzalez-Gurrola, L., Ramfrez-Alonso, G. (2018, September). ***Custom document embeddings via the centroids method: Gender classification in an author profiling task***. *In Proceedings of the CLEF 2018 Conference and Labs of the Evaluation Forum, 10-14 September Avignon, France. (Vol. 2125)*

- **López-Santillán, R.**, González-Gurrola, L. C., Montes-y-Gómez, M., Ramírez-Alonso, G., Prieto-Ordaz, O. (2019, September). ***An Evolutionary Approach to Build User Representations for Profiling of Bots and Humans in Twitter***. *CLEF 2019 Conference and Labs of the Evaluation Forum, 09-12 September Lugano, Switzerland. (Vol. 2380)*

A.1.3 Divulgation

- **López Santillán, J.**, González Gurrola, L. (2019). ***Determinación de Autoría de Textos***. FINGUACH. Revista de Investigación Científica Y Tecnológica de La Facultad de Ingeniería de la Universidad Autónoma de Chihuahua, 3(10), 16,17. Recuperado a partir de: https://148.229.0.27/index.php/finguach/article/view/289

A.1.4 Collaboration

- Casavantes, M., **López, R.**, González, L. C. (2019). ***UACh at MEX-A3T 2019: Preliminary Results on Detecting Aggressive Tweets by Adding Author Information Via an Unsupervised Strategy***. In Proceedings of the Iberian Languages Evaluation Forum (IberLEF Bilbao, Spain, September 24th, 2019) [127]

- Casavantes, M., **López, R.**, González, L. C., Montes-y Gómez, M. (2019). ***UACh-INAOE at HASOC 2019: Detecting Aggressive Tweets by Incorporating Authors' Traits as Descriptors***. In Proceedings of the 11th annual meeting of the Forum for Information Retrieval Evaluation (Kolkata, India, December 12-15, 2019) [128]

A.2 Other Publications

A.2.1 Journals

- **López, R.**, Gurrola, L. C. G., Trujillo, L., Prieto, O., Ramírez, G., Posada, A., ... Méndez, L. (2018). ***How Am I Driving? Using Genetic Programming to Generate Scoring Functions for Urban Driving Behavior***. Mathematical and Computational Applications (ISSN 2297-8747; ISSN 1300-686X for printed edition) Special Issue: Numerical and Evolutionary Optimization 2018, 23, 89. Emerging Sources Citation Index (Clarivate Analytics, 2019). Scimago Journal Rank (SJR), Impact Factor: 1.38, Computational Mathematics Q4.
https://doi.org/10.3390/mca23020019. [129]

- **J. R. López**, L. C. González, J. Wahlström, M. Montes y Gómez, L. Trujillo and G. Ramírez-Alonso, ***"A Genetic Programming Approach for Driving Score Calculation in the Context of Intelligent Transportation Systems,"*** in IEEE Sensors Journal, vol. 18, no. 17, pp. 7183-7192, 1 Sept.1, 2018. Indexed by the Journal Citation Reports (Clarivate Analytics, 2019), Impact Factor: 3.076, Q2. [130].

A.2.2 Congress

- O. Prieto-Ordáz, G. Ramírez-Alonso, L. C. González, **R. López-Santillán** and M. Montes-y-Gómez, ***"Brain Tumor Segmentation using an Encoder-Decoder Network with a Multiscale Feature Module"*** 2020 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), Ixtapa, Mexico, 2020, pp. 1-6, doi: 10.1109/ROPEC50909.2020.9258718. [131].

A.2.3 Collaboration

- Aguilera, Juan, González, Luis C., Montes-y-Gómez, Manuel, **López, R.**, Escalante Hugo J. (2020). ***From neighbors to strengths-the k-strongest strengths (kSS) classification algorithm***. Elsevier *Pattern Recognition Letters* Journal [132]. Indexed by the Journal Citation Reports (Clarivate Analytics, 2020), Impact Factor: 3.255, Q1, 2020 ISSN 0167-8655, https://doi.org/10.1016/j.patrec.2020.06.020 (https://www.sciencedirect.com/science/article/abs/pii/S0167865520302385)