Faculdade de Engenharia da
Universidade do Porto

Programa Doutoral em
Engenharia Informática

# Noise reduction and normalization of microblogging messages

*Author*
Gustavo Alexandre
Teixeira Laboreiro

*Supervisor*
Eugénio da Costa Oliveira
*Co-Supervisor*
Luís Morais Sarmento

February 5, 2018

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

*Dedicated to my loving family*

# Acknowledgements

First of all I would like to thank my family in loving recognition of their tireless support and understanding throughout this long endeavor. Such a work would not be possible without their help.

Further thanks are also warmly extended to my supervisor and co-supervisor who provided knowledge, help, guidance and encouragement through all these years.

Special appreciation is also due to the teachers who provided valuable insight and opinions, which greatly enriched this work and to whom the author feels indebted; particularly Eduarda Mendes Rodrigues and Carlos Pinto Soares. I would also like to express my recognition for the help provided by the members of the Scientific Committee.

For many days several colleagues accompanied me in this academic journey (but also a life journey), and who selflessly shared their time and friendship. To all of them I present an expression of gratitude.

The friendly department staff is not forgotten, specially Sandra Reis who always went beyond the line of duty to provide assistance in the hours of need.

The author would also like to make known the financial support that was provided through grant SAPO/BI/UP/Sylvester from SAPO/Portugal Telecom and Fundação para a Ciência e a Tecnologia (FCT) through the REACTION project: Retrieval, Extraction and Aggregation Computing Technology for Integrating and Organizing News (UTA-Est/MAI/0006/2009).

# Abstract

User-Generated Content (UGC) was born out of the so-called Web 2.0 — a rethinking of the World Wide Web around content produced by the users. Social media posts, comments, and microblogs are examples of textual UGC that share a number of characteristics: they are typically short, sporadic, to the point and *noisy*.

In this thesis we propose new pre-processing and normalization tools, specifically designed to overcome some limitations and difficulties common in the analysis of UGC that prevent a better grasp on the full potential of this rich ecosystem. Machine learning was employed in most of these solutions that improve (in a substantial way some times) over the baseline methods that we used as comparison. Performance was measured using standard measures and error were examined for better evaluation of limitations.

We achieved 0.96 and 0.88 $F1$ values on our two *tokenization* tasks (0.96 and 0.78 accuracy), employing a language-agnostic solution. We were able to assign messages to their respective author from a pool of 3 candidates with a 0.63 $F1$ score, based only on their *writting style*, using no linguistic knowledge. Our attempt at *identifying Twitter bots* was met with median accuracy of 0.97, in part due to the good performance of our stylistic features. *Language variant identification* is much more difficult than simply recognizing the language used since it is a much finer problem; however we were able to achieve 0.95 accuracy based on a sample of 100 messages per user. Finally we address the problem of reverting *obfuscation*, which is used to disguise words (usually swearing) in text. Our experiments measured a 0.93 median weighted $F1$ score in this task. We have also made available the Portuguese annotated corpus that was created and examined for this task.

# Sumário

Os Conteúdos Gerados por Utilizadores (CGU) nasceram da chamada *Web 2.0* — uma revisão da *World Wide Web* em torno de conteúdos produzidos pelos utilizadores. Exemplos dos CGU são as mensagens colocadas nos media sociais, os comentários em *websites* e os *microblogs*. Todos eles partilham algumas características típicas como a sua brevidade, serem esporádicos, diretos e *ruidosos*.

Nesta tese propomos novas ferramentas de pré-processamento e normalização que foram criadas explicitamente para superarem algumas limitações e dificulades típicas na análise de CGU e que impedem uma melhor compreensão do verdadeiro potencial deste ecossistema rico. Na maioria dos casos foi empregue aprendisagem automática, o que permitiu superar (por vezes de forma substancial) os sistemas usados para comparação. O desempenho foi calculado recorrendo a métricas padronizadas e os erros foram examinados por forma a avaliar melhor as limitações dos sistemas propostos.

Alcançámos valores $F1$ de 0,96 e 0,88 nas nossas duas tarefas de *atomização* (0,96 e 0,78 de exatidão), empregando uma solução sem qualquer dependência linguística. Conseguimos atribuir a autoria de uma mensagem, de entre 3 candidatos, com uma pontuação $F1$ de 0,63 baseada apenas no seu *estilo de escrita*, sem qualquer atenção à língua em que foi redigida. A nossa tentativa de identificar *sistemas automáticos* no Twitter recebeu uma mediana de exatidão de 0,97, em parte graças ao bom desempenho das características estilísticas mencionadas anteriormente. A *identificação de variantes da língua* é um problema muito mais complexo do que a simples identificação da língua, dado que é uma questão de granularidade mais fina; no entanto consguimos alcançar 0,95 de exatidão operando sobre 100 mensagens por cada utilizador. Por fim, abordámos o problema de reverter a *ofuscação*, que é empegue para disfarçar algumas palavras (regra geral, palavrões) em texto. As nossas experiências apontaram uma pontuação mediana ponderada do $F1$ de 0,93 nesta tarefa. Disponibilizámos livremente o nosso *corpus* anotado em português que foi criado e examinado para esta tarefa.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## Contents

When the movable type printing press gained presence in Europe, in the middle of the 15th century, many profound alterations took place in the society of the time. It changed sciences, religions, politics, the economy, and paved the way to the Renaissance [Eis05]. Before that, books were rare and precious items, inaccessible to the common man for fear of theft or wear. Clerics had the monopoly on the written word, and would dedicate themselves mostly to produce (inexact) copies rather than creating new works.

As books became much more accessible through mass production (a scribe's manual work was quite dear), people outside the literate elite finally had reasons to learn to read. Book production started to be dictated only by demand, meaning that Latin was no longer the language of choice for writing [Bra70, Eis05] (leading to a standardisation of the local languages [Coh61, Eis05]), and it became much easier for a person of the time to have their work published. Finally, knowledge and ideas could survive and travel longer in the form of many accessible tomes and avoiding the inevitable result of the concentration of documents [Eis05], of which the burning of the Ancient Library of Alexandria is the better known example.

Such significant revolution in information dissemination would not happen again until the end of the 20th century, with the World Wide Web. Internet publishing became simple enough that almost anyone was capable of creating their own website through the so-called *blog* services. As the information is now divorced from its physical medium, its cost of production, propagation and storage is further reduced, providing a more unencumbered circulation of knowledge and ideas. Millions of readers may now be within *instant* reach, and could also interact with the original author using special mechanisms (e.g. comments).

*Microblogs* — like other forms of User-Generated Content arising from the so-called "web 2.0" — bring this form of communication to a different level. Every reader/consumer can also be an author/producer. This "single role" means that all interactions are made in equal standing (as opposed to blog's author-moderator/commentator two class system), further fostering dialog. However, the most noteworthy aspect of microblogging is its ubiquity, as any cell phone can be used to access it. People can write and publish on a whim, leading to spontaneous remarks and informal, self-centred messages. This constant "up-to-the-minute" nature of microblogging is central to the interest of this new medium.

Twitter, the most well-known microblogging service reached perhaps its peak popularity during the 2017 US presidential election campaign and the months that followed it, due to its significant use by candidate and president Donald J. Trump, of which resulted some memorable moments, like the "covfefe" word[1] [Her17]. But, entertainment aside, it also raised some careful thoughts regarding the possible impact of a misused communication channel with such impact [Ber17]. This question is not limited to microblogs and does not affect only the popular users; hard questions are being asked regarding these newer forms of communication [Che18].

One advantage that microblogs have over many other forms of digital communication (like social networks that share many of the relevant characteristics we describe here) is that data is more easily available since most accounts are public. Twitter has been used to predict elections outcome [TSSW10], to detect epidemics [Cul10], to study earthquakes [SOM10], to analyse brand associated sentiments [JZSC09] and to perceive public opinions [OBRS10] among others.

What is special about User-Generated Content is the way in which it can mirror what is on the minds and souls of people. We can see that many publications are sporadic, unfiltered and not entirely thought through; they are also a habit for many users that post somewhat regularly (or semi-regularly) about their personal lives or simply the first thing they think

[1] That garnered ample news coverage from different angles. Just four examples:
https://www.nytimes.com/2017/05/31/us/politics/covfefe-trump-twitter.html
https://edition.cnn.com/2017/05/31/politics/covfefe-trump-coverage/index.html
https://www.independent.co.uk/life-style/gadgets-and-tech/news/\covfefe-donald-trump-tweet-twitter-what-mean-write-negative-press\-us-media-president-late-night-a7764496.html
http://www.nationalreview.com/corner/448148/covfefe-donald-trumps-twitter\-habits-white-house

of, by sheer impulse[2]. Most users share their publications with a close circle — just their friends and family (and for this reason feel safe to express themselves), but incidentally they also open the door to whoever may be interested — and several users do make an effort to increase the number of followers they have.

A number of research projects were developed at LIACC[3], SAPO Labs[4] and the REACTION work group[5], sharing the objective of gaining knowledge through the analysis of User-Generated Content, with particular interest in the Portuguese community. The work we present here is the result of our contributions to those efforts, and was developed with the aim of addressing particular needs that we felt were not being adequately addressed by the more usual methods. It focuses mostly on the pre-processing of messages, either removing or handling "noisy" tokens or identifying unwanted user accounts that could introduce "noise" into our message collection. Ignoring either of those possibilities could result in less accurate final results.

When we say "noise" we refer to irrelevant or meaningless data. We noticed that very few studies involving microblogs even mention pre-processing of the messages (the stage in the processing pipeline when noise is ideally handled), or how they handled situations containing noise. From what we are to believe that no significant attention was given to the matter. Is it because it is accepted that noise is inevitable? Is it because researchers are able to sacrifice a significant number of messages to errors among the many thousands available? Is it due to the lack of awareness of the impact of noise in the message processing? While we cannot prevent noise at the source, it is possible that some of it be converted into "signal" or simply avoided.

## 1.1 Research questions

Natural Language Processing, for a long time and for historical reasons, was dedicated to the handling of well-crafted corpora, such as literary or journalistic works. Those texts were carefully created: well structured, with strong coherence and usually focusing on one topic. The authors, abiding by the rules of writing, used proper syntax and grammar, and employed correct punctuation and capitalisation. The nature of most source material was either informative (like news articles), formal (e. g. legal documents) or

[2] Microblogs may be called a "write-only medium" by some, due to number of irrelevant posts that are of interest only for the author. These are often mocked as "what I had for breakfast" kind of posts. However, in our opinion such posts demonstrate how unfiltered microblog posts can be.

[3] Artificial Intelligence and Computer Science Laboratory, `http://www.liacc.up.pt`

[4] `http://labs.sapo.pt`

[5] `http://dmir.inesc-id.pt/reaction/Reaction`

literary. But today most text being produced is informal and randomly includes smileys, hashtags, mentions and URL. Being "in the moment" musings (a "write-and-forget" affair), users typically put in the minimum effort required for "good enough" posts where being understood is sufficient. The number of creators is also expanding rapidly — which only increases the variety in writing styles — making it more difficult to keep track of each author or knowing some background information to provide better context to their writings — things that could be done with printed authors.

Our work revolves around such questions pertaining to a new type of medium and aiming better understand what people are saying. We have grouped them into sets of two or three intimately connected research questions that we enunciate here.

### First set of questions

Generally we can say that text is comprised of words, numbers and punctuation; words are formed by letters, numbers are formed by digits and punctuation is made up by a small set of specific symbols. Yet current microblog messages can also take in new types of tokens which need to be treated with different rules, like smileys, mentions, hashtags and URL; each can contain letters, symbols and/or digits. At the same time the "old" tokens need to be handled differently, due to abbreviations and creative writing (e. g. "YES WE CAN / YES WEE.KEND"). Our first research questions, treated in Chapter 3 are as follows:

> Can a classification-based approach to tokenization of UGC clearly outperform a tokenizer based on rules specifically crafted for this purpose? If so, how much training data is required for achieving such performance? What amount of contextual information does the classifier need to obtain optimal tokenization performance?

### Second set of questions

While we may call it "creative writing", it is not clear if the structure displayed by microblog messages reflects some sort of personal signature or if it is more or less random. In other words, we would like to know if there are any idiosyncrasy in a text containing 13 or less words, and if users can find enough freedom to express themselves through their own voice or if we are facing linguistic anarchy without gaining anything. We researched the following questions on Chapter 4:

> What set of language-agnostic stylistic markers can we extract from microblog messages in order to differentiate between the writings of different people? How reliably could we attribute

> authorship among a small set of possible authors / suspects? And how large a sample would we require from each user in order to obtain solid results?

### Third set of questions

It is impossible for us to know who truly sits behind a given user account across the internet, but some accounts clearly show some noticeable pattern too strict for us to consider that a person is actively trying to engage with the community. Some are clearly identified as a news feed, others are dedicated to advertisement or simply spam (some times masquerading as a regular account).

We call such accounts "bots" (short for "robot" to indicate automated activity) and they are of significance because they can skew results from community analysis. For example, news feeds can give the impression that people are actually interested and tweeting about a given event when they are actually ignoring it. As an anecdote, when Pope Benedict XVI visited Portugal, in 2010, we noticed that about 1/3 of all tweets mentioning the event were published by news organisations. Therefore, the work presented in Chapter 5 was developed to answer the questions:

> What language-independent features can we use to identify non-human operated microblogging accounts? How dependable would such a system be (assuming the number of messages is no problem since bots are quite prolific)?

### Fourth set of questions

Just as we cannot be sure of *who* is behind an account, we also cannot tell *where* that person may be. We know that Twitter, for instance, has a user profile for each account, but that information

a) may be no longer true;

b) may be unrelated to their nationality (e. g. a tourist),

c) may be incomplete (e. g. a village or street name);

d) may be ambiguous (e. g. Lagos, Portugal or Lagos, Nigeria) or

e) may be intentionally misleading (e. g. Narnia, Hogwarts or the Moon).

In our projects we wanted to follow only users from Portugal. Selecting only users stating "Portugal" in their location would be too strict and leave many accounts out, while selecting users posting in Portuguese would be excessively tolerant due to the many Brazilian users using the same language. The resulting questions, that we treat in Chapter 6 are thus:

> Is it possible for us to reliably identify the nationality of a user based on the content (not metadata) of their messages? If so, how much text would we need for an accurate identification? What features would be the most telling?

By using only the content of the messages we may use this result on any environment, not just on microblogs.

### Fifth set of questions

Finally, while a lot can be said about swearing, it is unquestionably a form of expression, even if often treated with contempt. As such, it is somewhat common to find alternate forms of writing such words, either to disguise it (to make it pass through a filter) or to show a minimum of respect for the sensibility of other people who may be reading. We call this "obfuscation" and is not exclusive of dirty words — any kind of taboo word can be disguised, it just happens that profanity is nearly universal and in more common use than other special words.

Disguised words are most commonly ignored (and may introduce additional noise into the message). This can be interpreted as information that is being left on the table, unusable. Certainly a swear word is a swear word, and they should not be taken literally; but still they are not interchangeable and therefore contain value.

While profanity lists are common, tolerant and robust methods of recognising them are few. In Chapters 7 and 8 we try to address the following research questions related to this matter:

> What methods are used by the authors to obfuscate their swearing? Can we make use of this information to construct a deobfuscation mechanism that improves on the state of the art?

While we did not seek absolute and final proof on all our answers, we were satisfied with the conclusions we reached.

## 1.2 State of the art

As we stated previously, noise in microblogs is not uniform, and manifests itself differently based on the task being performed. This is reflected on the way researches typically approach noise: they either acknowledge its presence and multi-faceted nature in a shallow way when discussing the general view of their system [PPSC$^{+}$12, BDF$^{+}$13], or they focus their attention only on an individual task and address only a particular kind of noise that is relevant for their work [DMAB13]. This is understandable since the *noise* is not the main subject of either type of work.

We were unable to find a work dedicated to the in-depth study of noise in UGC, which leads, in part, the less usual structure of the present work, where we attempt to do just that. We discuss several forms of noise that was met performing multiple tasks in microblogs and other forms of UGC. In each chapter (starting with the third) we present the works we found relevant to that matter, and in this way we address then the state of the art in a more specific way.

## 1.3 Methodology used

After consideration and evaluation of each problem and research questions, we deliberate on the best way to approach it. The state of the art is consulted and compared to our solutions; occasionally we were unable to contribute with any improvements to it, but those situations are not described here.

Each work presented here was framed as a classification problem. This was not according to a plan, but it does mean that we present a very similar methodology and evaluation process across the work presented. The details can be found on the proper section in each chapter.

We collected relevant data in an unbiased way and performed the annotation that was required to produce a ground truth. This may then be used to create a model (training a machine learning algorithm for example) and for testing our classifiers. To ensure we work with a reasonable variance of inputs we shuffle the data and perform a N-fold cross validation test.

We use the standard evaluation metrics of *precision*, *recall*, *F1* or *accuracy* to quantify performance and, where possible, compare those results with state of the art. On some occasions we implement multiple solutions to provide a better comparison using the same data and evaluation method.

## 1.4 Contributions

Our first contribution lies on the usage of machine learning and a classification approach in the process of tokenization. This is a task that is traditionally performed through hard-coded rules, usually expressed through regular expressions. The improvement of the results were significant.

A second contribution lies on the study of the significance of the relation between the stylistics that can be observed in very short messages and their authorship and, in a more innovative way, determining the nationality of the author and human/robot accounts.

Another contribution we wish to highlight is our expansion of the Levenshtein edit operations with variable costs for the specific task of decoding disguised words in text.

Besides the more intangible knowledge resulting from our research and the solutions we describe in the present work, our contribution also includes the same tools we developed and used to perform our experiments. They are distributed at the SAPO Labs website through an OSI-approved free license.

While the license terms of Twitter makes it impossible for us to distribute the microblog messages datasets we worked with, we do include our binary models that we derived from them. This means that anyone interested in one of our solutions may start using it immediately and does not need to annotate data and generate their own models (but that possibility is always present).

Our work on obfuscation did not use data from Twitter, but from SAPO. We received official permission to distribute out annotated SAPO Desporto dataset, and do so hoping it will be useful for other researchers. More information can be found on Section 7.5.3, on page 119.

## 1.5 Thesis Structure

The following chapter will go into some depth explaining the relevance and challenges of microblogs (and User-Generated Content in general). It is meant to provide an introduction to User-Generated content, with strong focus on microblogs, which constitute most of the data we worked with.

As we stated, the problem sets we addressed are treated on their respective chapters, in the order in which we presented them. Each of these chapters is mostly self-contained but share a similar structure. They begin with an introduction and background to the problem and refer to related work. The methodology, experimental set-up and results obtained are then presented, discussed in detail and commented. They end with the conclusions we reached and with the identification of opportunities for improvement we would like to experiment with next.

To reiterate, on Chapter 3 we present our work on tokenization; on Chapter 4 we study the feasibility of performing forensic analysis on short messages; Chapter 5 describes our efforts detecting automatic posting accounts and Chapter 6 shows how we solved the problem of differentiating between the two major variants of Portuguese. Finally, in Chapter 7 we pour over the problem of word obfuscations with a particular emphasis on profanity.

At the end of the thesis we present a final chapter containing some final thoughts on the work overall and a set of future directions and related problems we would like to solve.

# Chapter 2

# UGC and the world of microblogging

## Contents

With the popular so-called "Web 2.0", User-Generated Content (commonly abbreviated as UGC, sometimes also called Consumer Generated Media) has become an integral part of the daily life of many Internet users. In fact, we claim it forms the central piece of many of today's most popular websites.

Wikipedia, Facebook, Youtube, Flikr, eBay, Twitter, Blogger and Craigs-List are examples of very different but popular websites. But all of them are dependent on the efforts of their users to continue to attract users and new contributions, and further enhancing their value.

Even if the website's "core business" does not deal with user's original work *directly*, many times it does so *indirectly* through comments, ratings, tagging, reviews, links and other forms. Amazon is one such example, as it encourages users to share their opinions, review products and suppliers, create lists of products that they think work well together, expose their wish-list and these ways create a sense of community.

User-Generated Content, while widely popular, is still understudied in many areas such as those here presented. That is to say that, based on our review of the state of the art, topics related to account selection (based on nationality or level of automation of the operation) or pre-processing of messages saw very little attention compared with trend analysis, the quantification of user influence or general opinion mining. While we try to provide comparison to studies in the same area that are somewhat comparable, in most occasions their results are not directly transferable to our work environment.

## 2.1 What is User-Generated Content

User-Generated content can be considered as any original or derivative product created by or through a system of peers, outside of their professional activity. This is a very broad definition, and covers from game "mods" and fan fiction to the "letters to the editor" newspaper column and movie reedits. In fact, scientific papers in a journal or conference proceedings can also be considered as User-Generated Content.

The disrupting factor of UGC comes from the "democratization" of the act of media distribution. It is now easy for one's work to reach a handful of friends, or many thousands or millions of other people that may be interested in what they have to say, to show or to share. This is sufficient encouragement for the massive amount of data that, for one reason or another, finds its way to Internet culture.

Characteristics frequently associated with UGC media productions are the low budget, the small work team, the very personal approach to the subject, and on occasion, the novelty factor that helps it stand out of the other works.

Currently, the popularity of UGC can, at times, overshadow that of professional products. For example, Wikipedia is currently more popular than Encyclopedia Britannica, and is also responsible for the end of life of other well known digital encyclopedias, like Microsoft's Encarta. Other collaborative efforts are currently gaining momentum, like Open Street Maps, and

Youtube has currently no real rival in the area of generic on-line video.

Professional media production is expensive, and for that reason, an effort is placed in quality. This implies the role of a supervisor, that is someone assigned to ensure that a certain minimum standard of excellence and rigor is consistently respected across the entire production. That standard, of course, is not set at the same level in different places. An editor has the responsibility of reviewing the texts before they are accepted for publication, and argues with the writers to make sure that the editorial line is followed. In other business, that work is performed by the producer, like in the music industry. When we talk about UGC, that responsibility is unassigned. Some times the authorship is unknown (for example, anonymous comments in some websites or discussion forum).

We can ask then, where can we find value in User-Generated Content? Roland Smart, a marketing consultant, probably summarises best in his blog [Sma09]. In his professional view, the value of UGC is that it:

- Democratizes sharing of information;
- Distributes content efficiently;
- Fosters dialog and diverse perspectives;
- Enables crowd-sourcing and problem solving and
- Provides large data-sets for analysis.

We are interested in a much more narrow subset of UGC, that has the following properties:

1. Is textual in nature;
2. Has tight space limitations;
3. Has a low participation cost;
4. Is of immediate nature;
5. Is ubiquitous; and
6. The data is public.

That is, short texts that anyone can produce from anywhere.

We can find this type of content in many places on-line, such as discussion forums, the comments sections in blogs and news websites, but microblogs are the best source, as they use no markup, provide a standard API (most of them), and provide abundant, up-to-date data.

At most, rules are set at a distribution channel (a website, usually) stating what is and is not accepted — usually to avoid legal complications —, and may dictate some sort of technical standard to be followed. But in general, the community is left to "sort itself out", through feedback (comments, votes, tags) and posting new versions — if at all.

## 2.2 A brief introduction to microblogs

Microblogs are a form of User-Generated Content that are used to follow (and propagate) the thoughts and doings of other people in a reduced number of characters (traditionally 140, but this is currently an arbitrary limit and therefore varies from platform to platform).

It is difficult to point out which system was the *first* microblog, as the service evolved organically from regular blogs. Some people trace its roots to personal diaries from the 18th and 19th century [Hum10]. But microblogging as we know it today certainly has its roots in IRC[1], IM[2] and SMS[3].

Both Twitter and Jaiku were at the genesis of microblogging in 2006 [Mad09, Bul09]. Twitter's story is quite well known. It was initially conceived as a system that provided updates through SMS, a medium that allows for 140 bytes per message[4]. To curb their SMS costs, Twitter creators decided to limit each update to only one phone message, and imposed the limit of 140 bytes [Mak09, Boi09][5]. Many other microblog systems adopted the same "140 characters or less" limit, even though they do not communicate via SMS. As of late 2017, Twitter expanded their limit to 280 characters.

While SMS microblogging is useful and appreciated by many users, it is often not the preferred form for interaction (if for no other reason, due to the costs involved). Many (if not all) microblog systems offer a web interface, RSS reading and native client applications (for one or more Operating Systems and/or mobile platforms). Some also provide a public API, allowing the development of non-official clients and applications. This includes native access via non-officially supported Operating Systems and microblog use for custom-made systems. For example, geolocation of stratospheric balloons[6], road traffic information[7] or pet monitoring[8].

Microblogging is not limited to just Twitter, but at the moment Twitter holds the lion's share of microblog users, relegating all other networks to market irrelevance, to the point that microblogging and Twitter are some times used as synonyms. There are, however, two caveats related to this.

[1] Internet Relay Chat is an open protocol for real-time multi-way text communication.

[2] Instant Messaging is a general term used for a form of real-time bi-directional text-based communication system, although some systems have extended it to allow for multi-way communication, and support live audio and video calling.

[3] Short Message Service is a component of cell phones' communication system used for the transmission of text.

[4] A total of 1120 bits that allows the encoding of 160 characters messages using a reduced 7-bit alphabet.

[5] Twitter messages are mostly encoded in UTF-8, thus their maximum length in characters depends on the characters used. If a messages is too long, is stored truncated, which may some times be unexpected, due to special characters.

[6] `http://spacebits.eu/page/project.html`

[7] `http://twitter.com/perth_traffic`

[8] `https://www.huffingtonpost.com/2010/02/15/puppy-tweets-tweeting-dog_n_462455.html`

First, QQ and Sina Weibo compete over the large Chinese market [con18a, con18c]. They are both giants , even if they are less well known outside that country. Second, there is Facebook[9], with its 500 million users.

While Facebook is a *social network* (and some people also claim that microblogs are technically social networks — a discussion that we shall not delve into), it is true that its "status update" feature is a form of microblogging, i.e. it allows its users to publish short text messages online and to read other people's updates. There are two key points that we consider to negatively impact Facebook's microblogging aspect: access to status updates from Facebook's users readable only by their friends, and access to the network requires access to a computer or smartphone, limiting a person's ability to participate at any moment. Therefore we are left with less meaningful and sporadic data that is more difficult to analyse.

Facebook status messages are limited to 420 characters. While people have experimented with status.net services, creating microblogs limited at 1400 characters, one of the key points in microblogging is the short message length.

On the other hand, Twitter officially claims 175 million users, that are responsible for 65 million messages per day [Gar10], accessing it through a multitude of methods [Wil10]. More recent rumours put those numbers at 200 million users and 110 million messages per day [Chi11]. This volume of information is quite impressive. Messages are in constant flow, and users can write from anywhere [LF09], resulting in up-to-the-minute updates and reactions from all around the world.

These are the reasons that turned our attention towards microblogs — and without loss of generalisation, Twitter — for our work: the large volume of data available for study, its significance, its nature and its foibles.

## 2.3 A few notes about Twitter

As stated above, Twitter is not the only microblogging service, but it was the first and is still by far the most popular. As it still holds the dominant position in microblog culture, we will talk about microblogging as seen through a Twitter user in this segment. The basic concepts of microblogging are also valid for the competing systems, sparing only some terminology — for example, in identi.ca a message is called a "dent" instead of a "tweet", and they use the term "resend" instead of "retweet".

Twitter is not limited to what is described here, but hopefully it will be sufficient to allow almost anyone to follow the rest of this document.

A Twitter *timeline* is simply the collection of messages that users publish, and is updated in real-time. A user also has their own timeline, containing only their posts. This timeline can be either public or private. If

[9] `http://www.facebook.com/`

private, then the user needs to grant access permission to each person that wishes to read it. Private timelines seem to be a minority in Twitter, and we intend to make no effort in accessing them.

Users are, of course, interested in consuming the most interesting timeline possible, and hopefully making a positive impact on others' timelines.

### 2.3.1 The network

If a user $A$ subscribes to another user $B$'s tweets, meaning that they want be able to see $B$'s updates, then $A$ is said to be a *follower* of $B$ (or simply that $A$ follows $B$). User $A$ can also have their own followers. On occasion, the word "followee" is used to denote someone that is followed, but it seems to be an unofficial term, as the Twitter website never uses it.

We can see that the graph of followers forms the backbone of Twitter's social network. However, and contrary to "regular" social networks, following someone in Twitter is not a mutual relation. $A$ follows $B$, but $B$ ignores $A$ is a very frequent pattern.

According to HubSpot's 2010 report [Hub10], Twitter users follow approximately 170 persons, and have 300 followers in average. But 82% of the users have less than 100 followers, and 81% follow less than 100 persons. This indicates that the social structure is very unbalanced.

Consulting Twitaholic's[10] numbers related to some of Twitter's top personalities (see Table 2.1), we can notice that some users have a disproportionate number of followers, and follow a great number of people in return. It is also apparent how this falls into Twitter marketing strategies. For example, Barack Obama's team is pulling a popularity trick (as in "I'm listening to you"), while Taylor Swift takes the opposite approach.

### 2.3.2 User interaction

In Twitter users can publicly interact with each other in three forms: mentions, replies and retweets. Twitter also provide a private "Direct Message" mechanism, but given that such data isn't public, we will not address it.

A Twitter *mention* is simply a message where a user's handle occurs in its body, prefixed by "@" — what is called some times a *@reference*. This symbol is used to disambiguate usernames from normal text. That way, "Twitter" means the microblog and "@Twitter" means the official Twitter account. For example:

> What I meant is **@twitter** could you please just let us keep old twitter? The new one kinda sucks, to be honest.
>
> OMG my **Twitter** account is 579 days old!!! <3 **@twitter**

[10] `http://twitaholic.com/`

Table 2.1: Some Twitter personalities and statistics from `https://friendorfollow.com/twitter/most-followers/`, retrieved on 2018-06-20.

| User | Role | Followers | Follows |
|---|---|---:|---:|
| Katy Perry | Pop singer | 109,595,659 | 216 |
| Justin Bieber | Pop singer | 106,535,597 | 310,606 |
| Barack Obama | Ex-president of the USA | 103,225,557 | 621,452 |
| Rihanna | Pop singer | 88,775,421 | 1,112 |
| Taylor Swift | Pop singer | 85,540,254 | 0 |
| Lady Gaga | Pop singer | 78,878,841 | 127,135 |
| Elen DeGeneres | Comedian | 78,098,165 | 35,757 |
| Cristiano Ronaldo | Footballer | 73,913,377 | 99 |
| Youtube | Internet media company | 72,303,826 | 1,026 |
| Justin Timberlake | Pop singer | 66,061,746 | 282 |

*Replies* are just like mentions, but the addressee appears at the start of the message. This indicates that the message is directed at that user (probably as a reply to something they said). However, the message is as public as any other. An example:

> **@twitter** I keep getting spam bot followed...fix this! :D

Both mentions and replies are gathered by Twitter, so that users can easily find out what is being said to or about them. A user can mention and reply to anyone, independently of following or being followed by that person.

Users can also propagate a message someone else wrote. This is akin to email's *forward* action, but in Twitter it is called a *retweet*. A retweet is identified by the letters RT and the reference of the quoted author. A copy of the message usually follows. Users can still edit the message, adding their comments, more often at the start of the message to clearly separate them. Here is an example:

> An inspiring must-read. Who knows what we'll accomplish in 2011?
> **RT @twitter** The 10 Most Powerful Tweets of the year: `http://ow.ly/3oX0J`

These interactions can all be combined. For instance, we can create a retweet of a retweet, or reply to a mention, or retweet a reply to a retweet to a mention.

### 2.3.3 Defining the context of a message

Another interesting and noteworthy constructor in Twitter is the *hashtag*. It is used to group messages related to a certain theme, and is created by using an hash symbol ("#") followed by a word. For example,

> My latest blog post: Gulf Seafood Safety in Question `http://wp.me/pZnhC-7X` **#gulf #oilspill #bp #corexit**, tweetin this again 4 the AM crowd!

> Snowbirds are coming to Myrtle Beach, **#Florida**, quelling fears of **#Gulf #oilspill** impacts: `http://bit.ly/gjYY5p`

We can see that in the second example the hashtags are also used as normal words; as opposed to the first example, where they are used more like "metadata" for topic grouping.

### 2.3.4 Takeaway ideas

Microblogging is a relatively new medium of communication that combines SMS with blogging (among other influences). It is meant to be short, simple, to the point, spontaneous, unstructured and informal. Twitter is the most influential microblog, and for this reason, the one adopted as the example.

It was Twitter that gave rise to two (user-created) constructors that are now common in may other domains, namely referring to other people by writing a name preceded with the symbol "@" (called a @reference or mention), and the hashtag, that consists in using the symbol "#" before the name of the tag. The simplicity of adding metadata as part of the message helped to popularize it.

As a tool that is "always at hand", microblogging quickly became part of many people's daily habits, extending their communication medium portfolio. It is used for big and important announcements and for daily and inconsequential chitchat, providing an all-encompassing medium, probably only comparable to face-to-face talking.

Processing microblog messages automatically can be very different from processing small excerpts of formal text (i. e. the typical type of text written by a person with care and attention to how they are expressing themselves, and we can find in books or newspapers). This can result in a number of problems for text processing systems, and we will analyze some of these issues in the following chapter.

## 2.4 How microblog messages can be problematic

Many of the tools that are currently used for linguistic processing can have their roots traced to the early era of computational linguistics. Kučera and Francis created the now-called Brown Corpus in the early 1960s [KF67], the first large corpus for linguistic analysis. At the time computers were much more limited. For example, the Brown Corpus was written in all-caps, with an asterisk indicating word capitalization [MS99]. Digital texts, while not rare, were not as pervasive as they are today.

Currently, we have a very large proportion of text being created digitally, as much more powerful computers are easily accessible to many people, lowering the bar for content creation. The Internet lowered the bar that restricted content distribution, especially when *Blogs* appeared. Today, we have an environment where large amounts of text, written by common people (i.e. available to anyone who wants it), are constantly being distributed, accessed and, hopefully, read. While no one denies the existence and relevance of what can be called the "clean text", carefully created by professionals, we claim that the more common problem (in text processing) has shifted significantly towards the disposable messages quickly typed by "average Joes".

The first problem researchers meet when trying to apply their linguistic tools on microblog messages is the large amount of *noise* [DH09, SRFN09]. Noise is the general term applied to outlier values that fall outside the model used for the data, and can interfere with the meaning of the message in an unwanted way. In our context, we call noise to any form of deviation between the original (microblog) message and the same message written in a clean and simple newspaper-like *form* — we are interested in dealing only with the lexical aspects at the moment, and will therefore ignore any syntactic or semantic factors.

Noise has a negative impact in the correct reading and understanding of the messages, for both humans and automatic systems. However, a person can more easily compensate using context, experience and cultural knowledge, and is the reason many of it is tolerated today. We follow with a brief discussion of many forms of noise present in microblogs.

### 2.4.1 Spelling choices

Alternate spellings are one of the most obvious differences between microblog text and "clean" text. This includes words that are misspelled either by accident or on purpose. For example, by using "Micro$oft", "Dubia Bush" or "Deusébio", the author is embedding a secondary message in the name: "Microsoft is greedy", "George W. Bush is a dumb Texan", and "Eusébio is a divine figure".

It is also possible to see the use of numbers instead of letters to type a word. Numbers can be chosen due to their graphic appearance, that resembles a letter, often seen in “l337 5p34k” (leet speak), or to their pronunciation, i.e., “2day” (today), “4get” (forget) or “pir8” (pirate).

Below we can see two consecutive messages from the same user, containing different spelling choices. The first message shows the misspelling a word as another similar word (the context hints that she meant “bores” instead of “boards”). The second message has errors due to the inclusion of an extra letter, and the removal of another one.

> SAM **BOARDS** ME *YAWN x

> OPPPS SPELLING **MISSTAKE** AND YOU SPOT IT OUT **EMBARRASING** :S

### 2.4.2 Orality influences

In general, UGC is strongly influenced by oral forms of communication. Especially in Twitter, as it contains mostly unprepared discourse. Messages tend to be short, and present syntactic structures that are usually simple, incomplete or even incorrect; we also find many sentences that are incomplete.

The vocabulary we find in UGC tends to be less rich since, for example, we find more broad words being employed (e. g. “thing”, “it”, “stuff”) and referring expressions (e. g. “she”, “her”, “that”, “this”, “here”, “there”). The oral influence can also be observed in the grammar (i. e. past perfect is rarely used), and in the strong contextual reliance that is required to understand the message correctly.

#### Phonetic spelling

One form of intended misspelling is the use of phonetic-like writing as a stylistic choice or cultural bias — for instance, when the author intends to give a “street culture” image. It can also be used when they try to call attention to someones’ speech pattern (for example, an accent or lisp). The following examples show what seems to be unintended errors and words misspelled on purpose (e.g. “ya” and “u c”):

> You can super **annoyin** but i **couldnt** live without **ya** <3

> @fakeforte **wassup** man **u c** m&m didnt win album of **da** year last **nite** or **wateva** instead justine bleiber won wtf **shulda** been **weezy** or **soulja** boy

#### Onomatopoeias, interjections and fillers

The orality influence also extends to the inclusion of as onomatopoeias, interjections and fillers. These special lexical categories show great variety across graphic variation (since there is no "official" definition of how they are written) and culture. For instance, the onomatopoeias "jajaja" and "jejeje" are popular in Spanish, as "hahaha", "hehehe" and "hihihi" are in European Portuguese, or "rsrsrs" and "kkkk" are in Brazilian Portuguese.

Fillers are used in speech to avoid an awkward silence when the speaker organizes their thoughts. Their presence in a written medium is a strong indication that the author wishes to carry across the full experience of spoken conversation.

Examples of their use follow.

> @VargasAlan **Cof cof** werever **cof cof**? **jajaja**

> @araeo **Err** ... that was 'writer's BLOCK', obviously. **Ugh**. I'm hopeless.

> @meganjakar **errrrrrr** not too sure. I don't know if you're invited..?

### 2.4.3 Non-standard capitalization

The purposes of capitalization vary according to language and culture. Some languages do not even use them, like Chinese and Japanese. When used, many times they are meant to highlight something, i.e., the start of a sentence, a proper noun, any noun (i.e. in German), an acronym, or the word "I" in English[11].

If the writer neglects standard or commonly accepted capitalization, they may also be negating very useful clues that help determine if a dot marks an abbreviation or a full stop. These clues would also be useful in hinting that "Barack" could be a correctly written proper noun absent from the dictionary, and not "barrack" misspelled. The following examples illustrate different situations. The first two show a consistent letter case choice. In the third example the user changes to all-capitals in the end of the message. The last example displays selective casing, where the author correctly capitalizes some words but not proper nouns.

> christmas convo: me- wat u want fa christmas her- u kno how i lik it http://bit.ly/ih8xsA

> SHOUTS TO DADDY K FOR THE KICKS HE GOT ME I LOVE EM AND SHOUT TO BOB SINCLAR I HAD A GOOD TIME @ THE TV SHOW LAST NIGHT !!

[11] "i" was a word that got lost easily in texts, and capitalizing it made stand out more. "I" became prevalent in the 13th and 14th centuries.

> Look, I don't mind following u Bieber freaks but keep those dam petitions OUT MY MENTIONS...

> Listening to **tyrone bynum** talk shit bout **barack obama**..... This nigga is in his own lil world....

### 2.4.4 Abbreviations, contractions and acronyms

As expected, when users need to fit their message into a small space, abbreviations, contractions and acronyms are frequently used to reduce the character count. The main problem arising from their use is the creation of non-standard forms when a more common one may already exist. In fact, the variety of methods available to cut characters in text naturally lead to this situation. For example, "lots of people" can be reduced to "lots of peop.", "lts of ppl", "lotz'a ppl", "lotta ppl", "LoP" and so on.

Many factors may contribute to the choice of less popular options over more established alternatives. For example, if authors really need to save more characters, they can opt for a more aggressive abbreviation. They can be typing from a device that makes the apostrophe more difficult to input. It could also be a question of writing style — to give a more oral tone to the message.

There is also the possibility that an author can think that the target readers will have more difficulty in deciphering a certain reduced form (e.g. it could collide with another word). For example, Wikipedia currently lists over 90 possible disambiguation alternatives for "PC", including "personal computer", "probable cause" and "program counter". Acronym Finder[12] lists 304, adding "pocket computer", "phone call" and "prostate cancer" among others. Hence, "not with PC" could be read as "not with probable cause" by someone with a legal background, and "not with prostate cancer" by a medical worker.

#### Abbreviations

In regular, careful writing, some words are very frequently written in abbreviated form, such as "etcecetera" ("etc."), "exempli gratia" ("e.g."), "Doctor of Philosophy" ("PhD"), "street" ("St."), "saint" ("St.") and "versus" ("vs."). In Twitter there are some standard abbreviations (e.g. "RT" for "retweet"), but they are uncommon as most "standards" in Twitter are *de facto* standards, that gained community support. For example, "#tt" may mean "trending topics", "travel tuesday" or "toilet tweet"[13], and "#ww" could mean "worth watching" or "writer's wednesday"[14].

[12] `http://www.acronymfinder.com/`

[13] `https://tagdef.com/en/tag/tt`

[14] `https://tagdef.com/en/tag/ww`

The large number of words that are abbreviated, and the number of possible abbreviations they can have lead to a fragmentation of the language. For example, “you” — already a short word — can be seen written as “ya” or “u” (as in the examples in Section 2.4.2).

One additional problem with abbreviations on Twitter is that many are written without the dot after them. For instance, “tattoo” is occasionally written as “tat”, which is also a well-known misspelling of the frequent word “that”. This problem is augmented when the abbreviated form of the word coincides with a regular word (e.g. “category” shortened to “cat”). Recognizing an abbreviated word is thus a challenge left to context.

Finally, a word can be subjected to alternate spelling *and* abbreviation, as in the case of “please” written as “plz” instead of “pls”.

In the following examples we can see “morning” written in the form “am” (an abbreviation for “ante meridiem”) and “because” shortened as “b/c”. It should also be noted that in the second example “ya” is used instead of “you” *and* “your”, showing how a user can fail at consistency *even in the same message*.

> @ryanhall3 saw **u** at my hotel this **am**. Didn't realize it was **u** until **u** were leaving. **Prob** a good thing **b/c** I would have asked **u** for a **pic** :)

> @alexandramusic hii!! How are **u**?? I've missed **ya**!! Omg i love **ya** song'bad boys' sooo much!!! Can **u** **plz** follow me!?? :) xx

### Contractions

Contractions can be very popular in general casual writing, and less in formal writing. As in microblogs the general style is very informal, it is expected that contractions are very frequent. For example, “I'm” is much more popular than “I am”, even if it saves just one character. However, users many times further reduce the expression to just “Im”, leaving to the reader the task of noticing the contraction.

Contractions are one of the many orality markers to be found in microblogging.

> **Yall im** finna stop **smokin**

> It annoys me when ppl say they **waitin til** the 1st to workout. U **aint** **doin nothin** today so get to **gettin**

### Acronyms

Acronyms are also very popular in microblogging, as they allow significant character condensation. An expression or idea can be encoded in just a few

Table 2.2: Some frequent acronyms used in microblogs.

| Acronym | Expansion | Acronym | Expansion |
|---|---|---|---|
| AAMOF | As a matter of fact | GF | Girlfriend |
| AFAIK | As far as I know | IANAL | I am not a lawyer |
| AKA | Also known as | IMHO | In my humble opinion |
| ASAP | As fast as possible | LOL | Laughing out loud |
| ASL | Age, sex, location? | OTT | Over the top |
| BF | Boyfriend | SMH | Shaking my head |
| BTW | By the way | SO | Significant other |
| FTW | For the win | TA | Thanks again |
| FWIW | For what it's worth | TLDR | Too long, didn't read |
| FYI | For your information | YMMV | Your mileage may vary |

letters, as can be seen in the examples in Table 2.2. Some have become very well known, like "LOL".

However, and contrarily to abbreviations and contractions, a good understanding of the language and the context is insufficient to decipher the acronym. It needs to be introduced or looked-up somewhere.

There are certainly thousands of acronyms in use, covering names, situations and quotes. Most are limited to specific contexts, but many are very popular in general use.

Popular acronyms are some times altered to change their meanings, as for instance, changing "IANAL" (I am not a lawyer) to "IANAP" (meaning "I am not a plumber"), or "IAAL" (for "I am a layer"), but it should occur only when very obvious from the context (or the "joke" is explained). Other types of changes affecting acronym are their use as words, such as "LOLed" or "LOLing" (meaning "laughed" and "laughing", respectively).

Finally, as by far the most common acronym in microblog use, LOL is used with many variations. A common one is the exaggeration of the acronyms to reflect an exaggeration in the laughing. For example, "LLLOOOLLL", "LOOOOL" or "LOLOLOL".

As was explained before, proper casing rules are often not followed, making acronyms more difficult to spot. We can see this in the first two examples of acronym use in Twitter:

> @LilTwist and @LilZa Need a **GF ASAP Lool** its upsetting seeing them lonely I need to help them find a nice girll :)

> @angelajames Reviewers are entitled to opinions. I don't reply to mine - reviews not for me, for readers **imho**. **Otoh**, do like good ones!

> @aarondelp **BTW**, Time Machine uses (**AFAIK**) **AFP** over **TCP**. You just need the **NAS** to provide the appropriate support.

### 2.4.5 Punctuation

Many microblog users do not regard punctuation as important. It is so frequent to find the reading of messages hindered by the misuse or absence of punctuation, that one can already expect it. Part of the reason may be due to cell phone typing, where some devices have only one key for all forms of punctuation, making the task of looking for the correct character tedious. Other users may see punctuation on such short messages as a mere formality. Or they are trying to save characters and/or seconds of typing (or, parsimoniously, they simply don’t care enough to bother typing them).

Consequently, readers must use grammatical and/or semantic clues to guess where one sentence ends and the next one starts. For example:

> Ouch Russell Brand put a photo up of Katy Perry up without make-up on (and deleted it) nice to know shes human! `http://plixi.com/p/66692669`

Other times microbloggers exaggerate on the punctuation, by repeating it, or writing it incorrectly, as shown below:

> OMG OMG!!!I just saw that @emilyrister followed me!!!WOW!You’re awesome!!I read your tweets since the beginning,it’s crazy.THANKS THANKS!!!!

> @So_Niked_Up22 What time is yo game**.?** N you right**..**it really aint nothin to do**.!**

### 2.4.6 Emoticons

In spoken conversations, a significant amount of the message is perceived through non-verbal language, such as speaking tone and body language. In textual conversations these communication channels do not exist. To address this problem, and reintroduce such information to the medium, people resort to typographic signals arranged in a way that resembles facial expressions. The most common of these is the very popular happy face “:-)”.

These constructions are added to informal conversations as a way of conveying a “tone” to the messages through happy (“:-)”), sad (“:-(”), winking (“;-)”), tongue out (“:-P”), crying (“:’-(”), surprise (“:-O”) or other aspects from a wide and varied range. The general name given to this class of graphic markers is *emoticon*, as it mainly expresses emotions through iconic representations.

Many variations of emoticons have been created, such as having the faces looking in the opposite direction (e.g. “(-:”) and with giving them a vertical orientation, instead of horizontal (e.g. “^_^”). These variations can also be altered to display a wide range of feelings (e.g. crying “T_T”, “)-':” and winking “^_~”, “(-;”). Emoticons are not limited to face-like images (i.e. a heart “<3”, and a fish “><>”). One simple example of smileys in use:

> @LadyGaGaMonst Oh, okay. **;D** BT is amazing. I'm so happy that Dancing On My Own got a Grammy nomination. **:D**

Emoticons are frequently an important part of the message, and not just a complement. Should they be removed, the reader could be left in a situation where they are unable to understand the message correctly, or at least not in its fullest. For example, in the simple message “It's already empty. :-)”, the emoticon adds something to the message, and changing it to another one — or removing it altogether — can result in significantly different meaning.

In ironic messages, what is written is not to be taken literally. Many times the author means the exact opposite. Emoticons provide a very valuable clue in this situations. For example:

> @Camupins Yeah, you're right. No one loves you anymore... So HAH! Suck that! **;D**

Emoticons are valid elements of microblog's general lexicon. This is a broader lexicon than the one traditional NLP applications expect to process.

### 2.4.7 Decorations

A number of microblog users believe they should highlight some parts of their messages. If a rich text format was available to them, they would use a bold, italic, underline, colour, blinking, sparkling, and/or another method to call attention. Since microblogs do not provide such features, there are only three alternatives available: using upper-case text, decorations or exploring Unicode characters.

The use of upper-case text has already been discussed in Section 2.4.3. It produces what is often called the “shouting” effect, as it gives the impression that the writer is raising the voice.

The decorations consist of one or more non-alphanumeric symbols that are placed around words or characters to make them stand out. These symbols are usually easy to distinguish from letters, as to not complicate the reading. Therefore, “.” or “:” are fine, but “#” or “$” are not as common. Some characters have been used in the Internet for many years for this purpose: “*bold*”, “/italics/” “_underlined_”. Here are two examples of decorators:

> Lea and ezra america. sitting in a tree. **k.i.s.s.i.n.g**

> @SpeakSeduction im feelin all sorts of. "emotions" haha damn . but shoot at least im not workin yay!! __*jigs*__

Decorations can also be used as “ornaments” in the message, without the intent of supporting words. These characters could be deleted from the message without affecting its meaning in any way, as they function only as beautifiers of spacers. For example:

> ∼CHECK IT OUT∼

It should be noted that the distinction between an unknown smiley and a decoration can be very hard to make some times. Here is a more complex example of the use of decorators:

> TYVM Love u back d(ˆ_ˆ)b #Joy RT @misslindadee __*¨)__
> God Bless!º≤≥**P::E::A::C::E ::&::**
> **H::A::P::P::I::N::E::S::S**!!!•*¨*LOVE YOU!!@SPARTICUSIAN

Finally, looking for more creative forms of expressing themselves, some users opted for exploring the Unicode character set supported by Twitter and other microblogs, and occasionally replacing letters by other symbols graphically similar, such as “a” by “@”, “E” by “∈”, “N” by ℕ, or “C” by “©”.

### 2.4.8 Obfuscation

On occasions, users wish to limit the number of readers that understand the full meaning of their message. There are many reasons for that: hiding profanity, avoid naming a person or company, or not wanting to be associated with some group or ideology by naming it.

There are many forms of obfuscation. The most common one consists of replacing one or more letters (usually not the first one) with another non-letter character. Vowels are usually the preferred letter to replace. Some users replace letters by graphically similar characters (e.g. an “o” by an “0”, or an “s” by an “$”). Another popular obfuscation technique is exchanging two consecutive letters from the middle of the word. Some examples of obfuscated text follows:

> Never understood why people edit they tweets by using words like **fcuk**,__sh*t__,
> **fuggin**,__bi**h__.What’s the point if you going to curse then curse

> ˝@Paris_Richie: ˝you a nothing __a**__ person , who does nothin __a**__ __sh*t__ , and ion want a nothin __a**__ nigga˝ !˝

There are some other text constructions that look like obfuscated text but have other meanings. One example is *globbing* (the use of a simple expression to define a pattern that encompasses a number of possibilities), such as in "*nix" to refer to Unix and Unix-like Operative Systems (Linux, Minix, Xenix, etc.). These occurrences are rare outside of the advanced computer area.

### 2.4.9 Special constructs

As email brought the idea of email addresses to the public, and the World Wide Web disseminated the idea of URLs, so does microblog bring with it a specific set of constructors, like the previously mentioned @references and hashtags. Even if for many people these are as common as phone numbers, for historical reasons (as was explained before), many linguistic tools still don't recognize any of them.

In the case of microblog-specific constructs, this is understandable, as they are limited to a specific environment; but others have become mainstream many years ago.

## 2.5 Conclusion

We have now seen how very informal writing can challenge automatic text processing. None of them are completely exclusive of microblogging, and can be found from most text UGC to SMS.

We can create an idea of the typical microblogging user based on some of the problems we listed:

- they can have some trouble in putting their thoughts in formal writing (they write as they talk), and while in some situations this may be intended, often it does not seem like it;
- they wish to avoid typing as much as possible, ignoring characters when they have no need to do so and disregarding letter casings; and
- they feel the need to emphasise things non-verbally, from stress (decorations) to non-verbal language (emoticons).

It would be an error to overgeneralize this idea, since some users do write in the most perfect form, but they are an unfortunate minority.

# Chapter 3

# Tokenization of micro-blogging messages

## Contents

The automatic processing of microblogging messages may be problematic, even in the case of very elementary operations such as tokenization. The problems arise from the use of non-standard language, including media-specific words (e.g. “2day”, “gr8”, “tl;dr”, “loool”), *emoticons* (e.g. “(ò_ó)”, “(=^-^=)”), non-standard letter casing (e.g. “dr. Fred”) and unusual punctuation (e.g. “.... ..”, “!??!!!?”, “,,,”). Additionally, spelling errors are abundant (e.g. “I;m”), and we can frequently find more than one language (with different tokenization requirements) in the same short message.

For being effective in such an environment, manually-developed rule-based tokenizer systems have to deal with many conditions and exceptions, which makes them difficult to build and maintain. That means that, while it would be possible to add support for mentions, hashtags and URL, other situations would be more difficult to handle. For example, the asterisks in "That is *not* OK!" (text enhancement) should be treated differently from the asterisk in "What a pr*ck!" (obfuscation) or in "Kisses! :-***" (smiley).

Hard-coded rules also have the effect of calcifying language-specific conventions and practices, making them less useful for use in a culturally diverse environment. Based on the requirements we mentioned, and the limitations our team was facing, we searched for a better solution.

In this chapter we propose a text classification approach for tokenizing Twitter messages, which address complex cases successfully and which is relatively simple to set up and to maintain. Our implementation separated tokens at certain discontinuity characters, using an SVM classifier to decide if they should introduce a break or not. We achieved F1 measures of 96%, exceeding by 11 percentage points the rule-based method that we had designed specifically for dealing with typical problematic situations.

Subsequent analysis allowed us to identify typical tokenization errors, which we show that can be partially solved by adding some additional descriptive examples to the training corpus and re-training the classifier.

## 3.1 Introduction

Despite its potential value for sociological studies and marketing intelligence [MK09], microblogging contents (e. g. Twitter messages and Facebook status messages) remain a challenging environment for automatic text processing technologies due to a number of foibles that are typical of User-Generated Content. Such environments tend to present a significant number of *misspelled* or *unknown* words and acronyms, and is populated with *media-specific* vocabulary (e.g. "lol") that is not always lexicalized in dictionaries. Additionally, it usually suffers from inconsistent use of capitalization in both names and acronyms, may have mixed languages and employ typographically constructed symbols (such as "=>" and ":-)"). Also, users often create new sarcastic words to better express themselves in a short and informal way (like "Dubya" to refer to George W. Bush). Some work has already been done in developing text *pre-processing* methods for homogenizing User-Generated Content (UGC), and transform it into text that is more amenable to traditional Natural Language Processing procedures, such as *contextual spell-checking* [Kuk92] and name normalization [ACDK08,JKMdR08].

The microblogging environment has some additional characteristics that influence the quality of the text. First, since messages are limited to 140–200 characters, text economy becomes crucial and thus users tend to produce highly condensed messages, skipping characters whenever possible (including *spaces*), and create non-standard abbreviations (such as “altern8”) — similar to SMS text [ANSR08]. Second, the quasi-instant nature of micro-blogging diffusion promotes conversations between micro-bloggers. Thus, messages tend to have a strong presence of *oral markers* such as emoticons and non-standard or missing punctuation. Finally, due to the almost ubiquitous access to a network, users can micro-blog from any place using mobile devices, which, for usability reasons, tend to constrain how users write messages and (not) correct them in case of spelling mistakes.

All these idiosyncrasies raise several obstacles to most natural language processing tools, which are not prepared to deal with such an irregular type of text. Therefore, some work has been done in developing text *pre-processing* methods for homogenizing user-generated content, and transform it into text that is more amenable to traditional Natural Language Processing procedures. These methods include, for example, *contextual spell-checking* [Kuk92] and name normalization [ACDK08, JKMdR08].

We focus on a fundamental text pre-processing task: *tokenization* (or symbol segmentation) as the initial step for processing UGC. The task consists in correctly isolating the *tokens* that compose the microblog message, separating “words” from punctuation marks and other symbols. Although apparently trivial, this task can become quite complex in user-generated content scenarios, and specially in micro-blogging environments for the reasons explained above. Besides all the tokenization problems that can be found in “traditional” text (e.g. acronyms, diminutive forms, URLs, etc.) there are many other situations that are typical in micro-blogging. As a consequence, mishandling of such highly non-standard language usage situations could compromise all further processing.

One more example, in regular text — such as in printed media — the slash (“/”) has a restricted number of use cases, and from a tokenization point-of-view it is unambiguous. In the microblogging context, a slash may also be part of a URL or a smiley. In fact, this kind of problem is also found in some specialized fields, such as the biomedical domain [TWH07], where some common symbols (such as “/” and “-”) may perform a function *within* the token, (e.g. "Arp2/3" is a protein name, and “3-Amino-1,2,4-triazole” is an organic compound). As their interpretation becomes more context-dependent, specialized tokenizer tools become essential.

Tokenization of microblogging messages thus may be quite problematic. The most common problem is users simply skipping white space, so the tokenizer has to determine if those spaces should exist. Table 3.1 has some illustrative examples. Message 1 shows the typical situation of the missing white space. While these situations could be fixed using simple rules, messages 2 and 3 exemplify other frequent situations where such a simple approach would not work. Messages 4, 5 and 6 show examples

Table 3.1: Examples of tokenization challenges. Key regions in bold font.

1 “the people at home scream at the **phone.and** complain when i **scream,like** ’ahhhhhhh’. **af,start** start start”

2 “Band in the style of Morchiba I just discovered: Flunk. Very good. Thank you **last.fm**!”

3 “i didn’t seee dougie **O.o** should I go to the doctor ??”

4 “@FirstnameLastname Really ? ? the first time I saw I was in **Frankfurt-GERMANY**, and this is the first time I saw Madri this way ,cuvered in snow.. kiss”

5 “@Berrador Benfica is not to blame that Vitor Pereira is irresponsible..But it’s bad for everyone. Notice the appointing of Duarte Gomes to the **fcp-scp**”

6 “Be a bone marrow **donor-Show** up in norteshopping until friday and unite for this cause.”

7 “What is worse, **coca-cola** without gas or beer without alcohol?”

8 “@aneite can i then schedule a **check-up** ? ( cc @firstlastname )”

9 “Today’s theories: **a)** if its in april, a thousand waters, then have rain until april **b)** if rain wets, then whoever walks in the rain gets wet -.-’ ”

10 “Have a good week, Guys **:o)**”

11 “@j_maltez Mother so young**!?!!?!! :X**”

12 “**5.8** earthquake in the Caiman. No victims reported”

13 “Earthquake of **6,1** felt today in Haiti.”

14 “The movie’s impressive numbers RT **@FirstnameLastname**: Ubuntu Linux is the key to Avatar’s success **http://bit.ly/7fefRf #ubuntu #linux #avatar**”

where the tokenizer should divide the emphasized text in three tokens: “Frankfurt - GERMANY”, “fcp - scp” and “donor - Show”. By contrast, in messages 7 and 8, the emphasized text should be considered as a single token. Cases involving parentheses also provide good examples of how tokenization may be complex.

Usually, parentheses make up their own token. However, when used in enumerations and *“smileys”*, the parentheses are part of the larger token (as in messages 9 and 10). Such cases may appear in other forms of text, but not with the frequency found in UGC. Other times users make a very creative use of *punctuation*, often overusing it (example 11). These occurrences, nevertheless, should not be confused with the very large number of smileys that populate the messages and which the tokenizer should keep together (as found in messages 3, 10 and 11). There is also a wide variety of complex cases including names of products (example 7) and numeric quantities (or dates) in non-standard format (examples 12 and 13).

Finally, there are also many microblogging-specific exceptions to traditional tokenization operations, such as *usernames* (@username), *hash tags*

(#hash_tag) and URLs that must be kept intact (example 14 illustrates these situations).

It would be possible to use an existing tokenizer and add new rules to cover these additional situations. One such tokenizer is *UPenn Treebank*'s tokenizer[1], which fails to correctly tokenize many of the above examples — for instance, it has no understanding of URLs or smileys, resulting in tokens such as "http : //bit.ly/123456" and ": - )". However, and contrary to *traditional* text, microblogging content is very irregular (with each author creating their individual writing style), and the token universe is *open*, due to the creative use of letters, numbers and symbols (i.e. characters other than letters, numbers and white space).

For the above reasons, many tokens are not in a dictionary. We cannot rely on compiling a list of possible emoticons, for example, and new ones continue to appear. Some can be quite elaborate, for instance: "8<:-)", "orz" (a kneeling man), "<(-'.'-)>", "(ò_ó)" or "(=^-^=)". The "orz" example in particular shows that simple heuristics are insufficient to identify all emoticons, as they can pass as an unknown word[2]. A more frequent example is the "XD" smiley. Without a compiled list or an adequate pattern to match them against, emoticons are difficult to accommodate in a rule system developed manually. This is not a small setback, as emoticons play a fundamental role in the interpretation of UGC-like messages, such as sentiment analysis [Rea05] or opinion mining tasks.

We propose a *text-classification* approach to the tokenization of microblogging messages. We train a classifier for deciding whether a white space character should be introduced *before* or *after* certain symbols, such as *punctuation*, *quotation marks*, *brackets*, etc. Compound words (such as "airfield") will never be separated.

This tokenizer is only concerned with the separation of tokens. We assume that other tasks normally associated with text pre-processing, such as the identification of compound multiword expressions, character transposition, text normalization, and error correction are to be delegated to other modules further down the processing pipeline (that can be performed in multiple cycles of operation).

For training the classifier we use a set of manually tokenized Twitter messages, which can be easily obtained, as the annotation task itself is relatively simple for humans — the annotator mostly deals with particular cases and does not have to think about the (not always foreseeable) impact of each rule they write. The key point is that adding more (and more diverse) training examples is an easier way to improve the performance than adding new rules [NY00].

[1] `http://www.cis.upenn.edu/~treebank/tokenizer.sed`

[2] "Orz" is also the name of an alien race in the fictional universe of computer game Star Control 2.

In this chapter we address the following questions:

- Can a classification-based approach to tokenization of UGC clearly outperform a tokenizer based on rules specifically crafted for this purpose? If so, how much training data is required for achieving such performance?

- Which features — both in diversity and amount of contextual information — does the classifier need to obtain optimal tokenization performance?

Our work is mostly done on Portuguese text, and that influences some of the decisions made when defining token borders. However, our methods and results are valid when considering most western languages with little or no adaptation.

The remainder of the chapter is organized as follows: in Section 3.2 we introduce some work that relates to tokenization or text pre-processing. In Section 3.3 we will describe the method used to tokenize UGC. In Section 3.4 we explain the rules for the processing of the corpus and why we made such choices. In Section 3.5 the experimental setup is described, explaining how we reached the results, that are then presented in Section 3.6. We analyze the most significant errors in Section 3.7, and then conclude with a summary of the results and directions for further work.

## 3.2 Related work

Being such a fundamental problem, tokenization is considered a solved problem on traditional contents. Maybe for this reason, there seems to be a lack of investment in developing new tokenization solutions for UGC, despite the fact that this type of content poses a completely new set of challenges for tokenization. Thus, in this section, we refer to works made on tokenizing text contents other than UGC, or on conceptually similar, yet different, problems.

A comparable work is the Tomanek et al. study of the tokenization problem on biomedical texts [TWH07]. The notation used to write *biomedical entity names*, *abbreviations*, *chemical formulas* and *bibliographic references* conflicts with the general heuristics and rules employed for tokenizing common text. This problem becomes quite complex, since biomedical authors tend to adopt different, and sometimes inconsistent, notations. The method proposed consists of using Conditional Random Fields (CRF) as text classifier to perform two tasks: (i) *token boundary detection* and (ii) *sentence boundary annotation*. For obtaining the corpus, the author gathered a number of abstracts from works in the biomedical fields, and extended it with a list of problematic entities. The set of features used for classification includes, for example, the unit itself, its size, the canonical form, the orthographical features, whether it is a known abbreviation and the presence of white space on the left or right of the analyzed point. After training the classifier they achieved a tokenization accuracy of 96.7% (with 45% of false positives).

Tomanek's successful method in biomedical texts cannot be reproduced when applied to UGC due to the different nature of the texts. Scientific articles obey general editorial rules. They impose a certain standard writing style, assure that the author is consistent in the language used (usually English), and that the text is easy to read. In UGC there is no review process, leading to many non-uniform messages (incorrect capitalization, for instance) with higher noise levels (non-standard punctuation, like ",,,"), multiple languages in the same short message (difficult to recognize due to misspellings and the small amount of text for statistical analysis), and as stated before, the open nature of the lexicon makes the "problematic entity list" strategy inviable.

Takeuchi et al. propose a method for solving the problem of identifying *sentence boundaries* in transcripts of *speech data* [TSR$^+$07]. The difficulty arises from the complete lack of punctuation in the transcripts, and also from word error rates of about 40%. The method, named *Head and Tail*, is a probabilistic framework that uses the information about typical 3-grams placed in the beginning (Heads) or in the end of sentences (Tails) to decide if there is a sentence boundary or not. The authors compare their method with a maximum entropy tagger trained on manually annotated corpora with boundary information. They also define a baseline that uses only the information about silence in the transcripts to decide if there is a boundary. Results show that the method proposed outperforms both the maximum entropy tagger and the baseline. Moreover, results improve further when the Head and Tail method is configured to use silence information and additional heuristics to remove potentially incorrect boundaries. Furthermore, they showed that their boundary detection method lead to significant improvements in subsequent morphological analysis and extraction of *adjective-noun* expressions.

We believe that the tokenization task we are addressing is, in certain ways, similar to the word segmentation problem in Chinese [TWW09]. Although tokenization of UGC is expected to be simpler than the word segmentation in Chinese, they both require making use of contextual information for deciding whether two symbols should be separated or not. Additionally, and contrary to word segmentation in Chinese, tokenization approaches in microblogging environments cannot make a solid use of dictionary information, since a very large number of "words" are not lexicalized. Our tests estimate that nearly 15.5% of tokens containing only alphabetic characters are not in a standard lexicon[3].

[3] The Portuguese GNU Aspell at `http://aspell.net/`

The most successful approaches to the tokenization problem use machine learning techniques. For example, Tang et al. compare the performance of three classification algorithms, namely (i) Support Vector Machines (SVM), (ii) CRF and (iii) Large Margin Methods, for achieving tokenization [TWW09]. Results show that automatic classification methods applied to text can achieve relatively high F-measures (95%). This led us to believe that machine learning approaches can also be applied with success to the task of tokenizing microblogging messages. On the other hand, the task of *word chunking*, i.e. grouping words together to form lexical or syntactical compounds, can be seen as special type of tokenization where each character is used as a "token". Again, machine learning approaches have proven successful. For example, Kudo and Matsumoto used several SVMs with weighted voting [KM01]. Their results also show that accuracy is higher, regardless of the voting weights, compared with any single representation system. Since the current problem is conceptually simple, a single classifier should be enough.

## 3.3 A method for tokenizing microblogging messages

A token is an atomic symbol used in text processing. Its definition is very dependent on what role it will play in the following processing pipeline. For instance, when tokenizing text for machine translation, it may be preferable to identify frequent expressions as tokens and not just single words. This "flexibility" is reflected in different results when comparing different tokenizers [HAAD$^{+}$98], that may not be simply interchanged.

Our intent is to perform information extraction on microblogging messages. For that it is fundamental that we first correct the frequent misspellings made by the users, and then normalize the inconsistent text. If we opt for a "traditional" tokenization process, we may split tokens that are difficult to piece back together at a later time. (E.g. "alternate" typed as "altern8", or "I'm" typed as "I;m".)

We define that each token is an element that can *later* be considered a unit of processing, for example: a word, a punctuation block (e.g. "!?", ","), a URL, or a microblogging-specific mark such as a user reference (e.g. @PeterP).

Tokenization is achieved by adding a space character in special *decision points* of the microblogging message. These points surround all non-alphanumeric characters (excluding the white space). In the following example, we use the symbol "||" to indicate decision points where white space should be inserted, and mark with a "|" those that should be left intact (i.e. white space is already present or adding a space character would break the token).

"Oh||,| if I was in Sampa||.|.|.| I would certainly go||!| |"||Beat it||"|

> Flash Mob||,| 25|/|01||,| at 14h||,| in Anhangabaú Valley||.| |(||via |@|terciors||)”

The tokenization method that we propose can thus be stated as a *binary classification problem*: can we train a model to determine if a certain decision point should have a space character inserted (or not)?

At this point, the question of an adequate training corpus arises. In Section 3.5.3 we describe how we annotated microblogging messages to create such a data source.

One aspect that is also worth exploring is the performance of the classifiers with different sizes of corpus. This is an indication of the amount of effort required in the manual annotation task to be able to achieve certain results.

### 3.3.1 Extracting features for classification

The first step in our method is defining the features that we use. These features are based on the properties of each *character* that composes the message. Therefore we can say that we work at *character level*. Every character is described by at least one feature.

All features used are binary (they are either present or absent), and describe different aspects of the characters (e.g. nature, function and purpose). No effort is made to reduce the number of features — it is up to the classification algorithm to identify the relevant features. In total we created 31 distinct features. Some examples follow:

- Character nature: alphanumeric, alphabetic, numeric, punctuation, symbol, space, etc.

- Type of alphabetic characters: upper case, upper case without accent, letter with accent, vowel, etc.

- Type of symbol and punctuation: bar, dash, monetary symbol, closing symbol (i.e.“(”, “>”, “}”, “]”), opening quotation (e.g. “"”, “«”), accent (e.g. “^”, “~”), arithmetic operator, smiley nose, etc.

The window of characters for which we generate features extends from the relevant decision point in both directions. This means that when using a 10 character window, the classifier can base its prediction on the features of the previous 10 characters, and the next 10 characters. The size of the window is one of the parameters of the process. One should take into account that using all characters in the message would greatly increase the feature space, and thus the computational requirements. Even worse, it could also worsen the tokenizer's results. This could happen as the classification algorithm may infer incorrect patterns due to insufficient number and variety of examples covering the extended context.

Table 3.2: Feature creation to the left and right of the first decision points (the first periods) of part of the text "if I was in Sampa... I would certainly go!".

| Left (pre) | | | | | Right (pos) | | | | |
|---|---|---|---|---|---|---|---|---|---|
| … | a | m | p | a | . | . | . | | … |
| … | 4 | 3 | 2 | 1 | 1 | 2 | 3 | 4 | … |
| … | m | p | a | . | . | . | | I | … |
| … | 4 | 3 | 2 | 1 | 1 | 2 | 3 | 4 | … |

The distance between the character under evaluation and the current decision point is taken into consideration, as well as if it is located to its left or right. For example, "if i was in Sampa... I would certainly go!" in the post quoted above would be interpreted as displayed in the Table 3.2 at the first relevant points.

The relative position of the character and the name of the feature are encoded into the features themselves. For example, the feature that we used called "`char\_is\_alphanumeric\_pre-3`" says that an alphanumeric character is found 3 characters to the left of the decision point.

Both positional and character information are required to address certain vocabulary-specific cases that are usually problematic in tokenization, like frequent abbreviations (for example, "Dr." and "etc.") and URLs ("http://"). The relevant feature in these cases is the *literal* feature. It represents each individual character, and provides the information "this is a "D" character", or "this is a "$" character".

For each decision point the number of features is usually large. For example, the simple letter "a" results in 9 features, meaning that it is easy to reach 100 features or more, even with a modest window size. Each character can be described by a number of features that is close to 180 (one for each literal character, plus the 31 category features we defined). Even with a small feature window size, the dimension of the *feature space* is very large.

## 3.4 Tokenization guidelines

The rules that govern our tokenization will have a big impact in later processing tasks. Subtle decisions at this stage can lead to very different results down the NLP pipeline [HAAD$^+$98].

A set of annotation guidelines was produced to ensure a good level of consistency during the manual annotation of the training corpus. These are summarised below to describe the corpus creation effort — the problems found and the decisions made. As a general rule, to avoid compromising the subsequent processing phases, it is essential not to "break" some "malformed" constructions, and lose the opportunity to correct them in later stages. Hence, we focus on what the author intended rather than what they actually typed.

We have emphasised that, in tokenization, the same set of rules may not work correctly in all situations. This also applies here. The set of "rules" enforced by our model are the ones that we feed to it. In our experiments we trained it with the purpose of supplying results adequate for text processing (as was our intention of use). When using our method to tokenize messages for deobfuscation — a much more specific situation that had no weight here (see Section 8.3.2) —, we found that our method showed no significant improvement over the results of the baseline in isolating those particular cases [LO14a]. Odd situations should be included into the training set — a constantly updated model may be desirable.

### 3.4.1 Word tokens

Word tokens are defined in a loose way, due to the nature of the messages. For example, "2day" is treated as an equivalent to "today", and both are equal *word* tokens. Twitter usernames (@twisted_logic_), hash tags (#Leitao2010) and hyphenated compound words ("e-mail") are also considered regular words/tokens. We commonly see the dash as a replacement for a word in certain circumstances where the length of the message is relevant. For example, with "Egypt–Nigeria" the author means "Egypt *versus* Nigeria", or "from Egypt *to* Nigeria", and therefore it should be tokenized as "Egypt – Nigeria". Also, acronyms are traditionally written with a dot, as in "U.K.", and need to be recognized as one unit. In case of abbreviations, the dot is also considered part of the word.

Another situation is the contraction used in informal oral speech, where the apostrophe is used to indicate that one syllable is unpronounced. This is seen in English, as in "li'l" or "lil'" meaning "li**ttle**", or in Portuguese where "'tava" stands for "**es**tava". Once again, the token is considered as atomic, since the apostrophe is meaningless alone in such a situation. However, the apostrophe is also used to combine two words, as in "I'm". In this situations, it should be tokenized as two separate tokens (in the example: "I 'm" — referring to "I am", only that we intend to modify the message only by introducing spaces).

Special care is taken with words that are typed incorrectly, as we expect them to be a frequent occurrence. Handling them properly at this stage will help in correcting them more easily at a later stage. The general rule is treating the string as if it was correctly written. A common error is the user pressing the key next to the correct one. For instance, "I**;**m" instead of "I'm" should be split as two tokens; while "im**[**ossible" instead of "im**p**ossible" should be kept as-is. Another problem with keyboards is when they are misconfigured (set to an incorrect layout). A user expecting to type "for**ç**a" in a Portuguese layout would produce “for;a” if the keyboard would be set to US.

Other difficult situations are deliberately created by the user. For instance, when they cannot find the symbol they want on their keyboard. One example is writing the “©” symbol as “(c)”. Typing accented letters in some foreign keyboards is a more complicated problem. The apostrophe is a known way of signaling acute or grave accents in these situations; for example, typing “ate'” meaning “at**é**”. One final situation happens when a user changes one or more letters in their text to obfuscate it, as a way to avoid censure (e.g. “sh*t”). Some of the occurrences described here are not very frequent, but they illustrate the diversity of problems being addressed.

### 3.4.2 Numeric tokens

We decided to separate the numbers from the units they represent, as some people write “1€” and others opt for “1 €” or even “€1”. Dates and times, that possess a more complex representation, are preserved as a single token whenever possible.

The comma is used as the decimal separator, and the dot as the thousands separator in Portuguese. But the opposite is just as common due to foreign influence. (See examples 13 and 14 of Table 3.1). A normalization process will solve this problem.

Another problem occurs when dealing with ordinals, where numbers and letters or symbols are concatenated. For example, in English UGC “$1^{st}$” is typically typed as “1st”, while in Portuguese it is written as “$1^{o}$” or “$1^{a}$”, and it is often found written as “$1.^{o}$”, “$1.^{a}$”, “1o” or “1a”. No space character should be inserted in any of these situations, as it represents a single word, and information could be lost (e.g. interpreting “st” as a short for “street” or “saint”).

### 3.4.3 Punctuation and other symbols

When dealing with punctuation, end-of-sentence symbols (“.”, “. . . ”, “!” and “?”) are kept grouped as a single token. The other punctuation characters and symbols create their own token, except in the rare instance where they are duplicated. In this way, “--,” is transformed into “-- ,”, that is, the dashes are kept grouped, and the coma is another atom. This rule is used for all non-punctuation symbols as well.

There are a few exceptions to the rules described above: URLs and emoticons/typographical symbols (e.g. “=>”) are not broken-up, as they would otherwise loose their meaning. The same applies to enumerators such as “a) b) c)”.

As an illustration, we present the original message:

> "I'm off to sleep a little bit,,uahua.. .but I'll be back to see the soap opera TO LIVE LIFE **=D** I loove.. rs....!!! Kisssses to those who like them **=D** rsrsss"

And the correspondig tokenized message:

> "I'm off to sleep a little bit ,, uahua .. . but I'll be back to see the soap opera TO LIVE LIFE **=D** I loove .. rs ....!!! Kisssses to those who like them **=D** rsrsss"

## 3.5 Experimental set-up

Our experiment is based on the comparison of the performance of a classification-based approach with that produced by a regular expression based method that will be used as a *baseline*. This rule-based method is inspired by the UPenn Treebank tokenizer, but follows our rules of tokenization previously defined in Section 3.4, and consequently is better suited for UGC than UPenn's.

### 3.5.1 The tokenization methods

We opted for using SVM [Joa98] as our classification algorithm, which were found to be extremely robust in text classification tasks, and can deal with the *feature space* we have (in the order of $180^N$, where $N$ is the number of characters in the feature window). In addition, they are also binary classifiers, as the decision is always between inserting or not inserting white space. We used an "off-the-shelf" implementation of the SVM classifier (SVMLight[4]).

To determine the best kernel to use with the SVM, simple preliminary tests were executed. The second degree polynomial function outperformed the default linear kernel by a small but significative amount, and therefore is used. However, an exhaustive search for optimum parameters for the SVM falls outside the purpose of this work, and the remainder of the parameters were left at their default values.

We wish to investigate how this classification approach behaves according to two variables: (i) the dimension of the corpus and (ii) the size of the feature window.

Our *baseline* algorithm works by trying to isolate tokens that belong to one of a series of categories, each one defined by a regular expression. They are, in order of definition:

1. URL, identified by the "http://" or "www.";

[4] http://svmlight.joachims.org/

Table 3.3: Tests generated from an example message for both scenarios.

| Tokenized | `@PeterP Yes ;-) ...` |
|---|---|
| $S_{all}$ | `@PeterP Yes;-)...` |
| $S_{one}$ | `@PeterP Yes;-) ...`<br>`@PeterP Yes ;-)...` |

2. a block of alphanumeric words starting with a letter (like a regular word);
3. a date, as “2010-07-08” or “25/12/2009”;
4. a time, such as “4:15” or “18h15m”;
5. a number, for example, “-12,345.6” or “$2^{\text{o}}$”;
6. one of about 25 popular smileys and variants;
7. a block of punctuation characters;
8. a user reference (“@john”);
9. a hash tag (“#true”);
10. a block of unexpected symbols.

### 3.5.2 Evaluation scenarios and measures

Both methods (classification and rule-based) are evaluated under two scenarios, that we consider complementary. In the first scenario, $S_{all}$, we remove *all* white space adjacent to each decision point, joining some tokens together (but not sequences of words). In the second scenario, $S_{one}$, the test messages have been correctly tokenized except for *one* white space that is removed. We generate one test for each white space symbol excluded in this scenario.

We will use the tokenized message “@PeterP Yes ;-) . . . ” as an example. It has 8 *decision points*: “@|PeterP Yes |;|-|)| |.|.|.”. Only two of the three spaces will be removed, as one is not adjacent to a *decision point*. The test messages produced from this example text are shown in Table 3.3.

When evaluating the tokenization results, for each decision point, a *True Positive* ($TP$) is a space character that is correctly inserted, and a *True Negative* ($TN$) is a space character that is correctly *not* inserted. A space character inserted in the wrong place is considered a *False Positive* ($FP$), while a missing one is a *False Negative* ($FN$).

The performance can thus be determined using Precision ($P$), Recall ($R$), F1 and Accuracy ($A$) measures calculated as

$$P = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$
$$R = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$
$$F1 = \frac{2PR}{P+R}$$
$$A = \frac{TP+TN}{TP+TN+FP+FN}$$

These values are all obtained using a 5-fold cross validation process.

We can see that both scenarios address different aspects of the problem. In the first scenario all spaces around *decision points* have been removed, so the tokenizer has to put them back. This scenario tests the *Recall* of the tokenizer. The chance of introducing incorrect white space is relatively low, and adding back spaces is important. The scenario $S_{one}$, where only one space next to a *decision point* was removed, tests the *Precision*. Any other white space added is incorrect, as there is *exactly one correct classification*. The situations typically found in UGC lie somewhere in between these two scenarios.

When testing 2500 messages, $S_{one}$ generates 10,838 tests, resulting in an average of 4.3 different versions generated from each test message. However, the number of possible decision points in $S_{one}$ is lower than those in $S_{all}$ (by almost 60%), as we ignore *decision points* next to white space.

As a way to compare the difficulty of both scenarios, we run a trivial tokenization method to the problem: always inserting a space character at every decision point. This will not always provide the correct tokenization, but is a simple and predictable method. Its performance illustrates how complex it is to solve each scenario.

Both situations are generated for each manually annotated example. We remove all spaces from the decision points when testing the first scenario, and produce a set of many different instances, where each one is missing a single token boundary (totaling the number of decision points) for the second scenario. In this way it is trivial to compare the result of the tokenizer with the manually annotated corpus.

### 3.5.3 The corpus creation

The data used in the classification consists of a manually annotated corpus with 2500 messages from Twitter users in Portugal, collected using the SAPO[5] broker service. These messages were randomly selected from

[5] `http://www.sapo.pt/`

6,200,327 posts collected between 2010-01-12 and 2010-07-08, from a total of 93,701 distinct user accounts. The vast majority of the messages are in Portuguese, but there are traces of other languages. To split the workload, these messages were manually tokenized by 5 annotators using the rules stated in Section 3.4, at an average rate of 3 messages per minute. To ensure a good level of consistency, the entire set of examples was later reviewed by the first author. The testing examples used on both scenarios are automatically generated from this corpus, that has all the spaces correctly introduced, by removing the space characters in (one or all) decision points.

## 3.6 Results and analysis

Figure 3.1 illustrates the performance (F1) of the SVM classifier in both scenarios ($S_{all}$ and $S_{one}$) as a function of the dimension of the training corpus. We used a feature window of size 10 in this experiment. The rule-based system simply ignores its training data. We can see that with just 100 training examples the SVM classifier already outperforms the rule-based tokenizer significantly, as expected. Adding more examples consistently improves the SVM's results. The rule-based approach would only provide better results if we added new rules — hence its almost stagnant F1.

It can also be observed from Figure 3.1 that the results for scenario $S_{all}$ are always higher than those obtained for $S_{one}$. The best F1 for the same experiment performed by the trivial approach (always inserting white space) are 0.672 for $S_{all}$ and 0.273 for $S_{one}$. So we can appreciate the relative complexity of each task. If we were to try a trivial tokenization approach — always inserting a space at every opportunity — we would obtain F1 values of 0.672 and 0.273 for $S_{all}$ and $S_{one}$ respectively. This trivial approach illustrates the relative complexity of each scenario. To illustrate the relative complexity of each scenario we tried a trivial tokenization approach — always inserting a space at every opportunity. This oversimplified method obtained F1 values of 0.672 and 0.273 for $S_{all}$ and $S_{one}$ respectively.

In Figure 3.2, we can observe the results of varying the feature window's size. Scenario $S_{all}$ is less influenced by the size variation of this window than scenario $S_{one}$. It can also be seen that the window of size 10 used in the previous experiment is *not* optimal, as the highest performance lies between 3 and 6 characters. Window sizes larger than 6 characters (to the left *and* the right of the *decision points*) show a decrease in performance. This can be justified by the sparsity of the features being generated, taking into account the size of the training set.

Significant values for feature window sizes considering F1 and accuracy are shown in Table 3.4. These include the highest F1 values as well as the performance obtained when using a 10 characters feature window

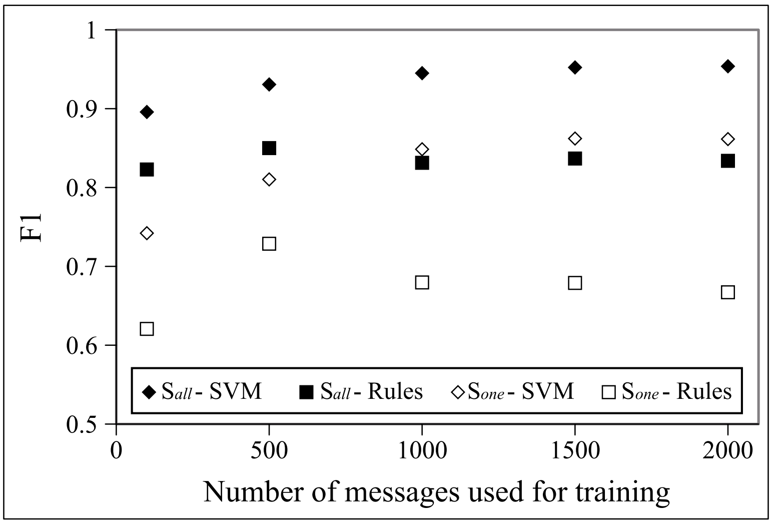


Figure 3.1: F1 vs. training corpus size for SVM classifier for $S_{all}$ and $S_{one}$, and analogous test for the baseline rule-based system.

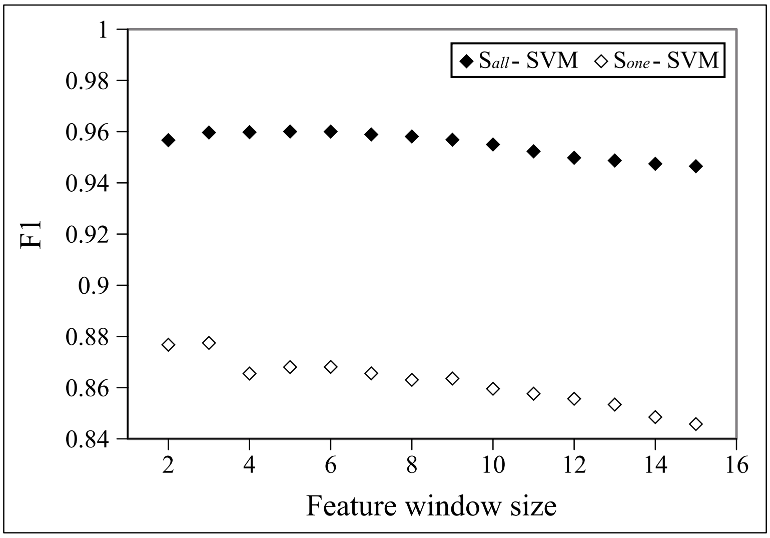


Figure 3.2: F1 vs. feature window size in characters for SVM classifier, trained using 2000 messages for $S_{all}$ and $S_{one}$.

Table 3.4: Feature window sizes for $S_{all}$ and $S_{one}$, by F1 and accuracy values

| Window | $S_{all}$ | | $S_{one}$ | |
|---|---|---|---|---|
| size | F1 | A | F1 | A |
| 3 | 0.9597 | 0.9622 | **0.8775** | **0.7817** |
| 5 | **0.9600** | **0.9626** | 0.8680 | 0.7668 |
| 10 | 0.9550 | 0.9583 | 0.8596 | 0.7537 |

used to evaluate the effect of corpus size. It is not easy to compare our results with related work, since most tokenization projects (reviewed in Section 3.2) have different goals, and do not deal specifically with user-generated content. Even so, we claim that the results that we obtain in our task are at the same level as those presented in other works, such as 96.7% accuracy in the tokenization of biomedical texts [TWH07] and 95% F-measures in the tokenization of texts in Chinese [TWW09].

## 3.7 Error analysis

We will describe the most significant errors made by the classification-based tokenizer when trained with 2000 messages (500 testing messages), with a feature window of size 3, as it provided the best general results.

Table 3.5 shows the characters most frequently associated with tokenization errors. As expected, the most problematic characters are those that should be tokenized in different ways, depending on their context. Those challenges have been presented in the discussion of Table 3.1 and presented in Section 3.4, so we were not surprised with this result. The errors that we found are related with the dificulties that we mentioned. Every character in the table can be tokenized "alone" in some situations, and as part of a larger token in other contexts — for example, when part of an emoticon (even if some characters are less frequently used in this way).

We can see that there is no direct correlation between the frequency of the characters and their tokenization error rate. For example, the comma is not even present in the table, while 748 out of our 2500 messages (28.7%) have at least one comma. This can be justified by the small number of functions associated with the comma. At the same time, the apostrophe is only present in 17% of the messages, and is processed incorrectly more often than correctly. This, however, is not true with the other characters in the table.

One of the most surprising error situations arises from the use of "." as a mistyping of "-", that was more frequent than we expected. These keys lay side-by-side on the Portuguese keyboard layout.

The other unexpected result comes from the difficulty of processing the apostrophe correctly. Part of the reason derives from the many languages found in the messages. The better known cases such as the “'s” in English, the “l'” in French and the “d'” in Portuguese and Spanish require more examples than those available to infer the correct classification pattern. In addition, the use of the apostrophe as a replacement for an accent or part of an emoticon is also more frequent than we anticipated. This situation also reinforces the need for more examples that address uses of this character.

Even though the values in Table 3.5 are biased towards the *False Positives*, overall values in scenario $S_{all}$ were balanced.

The dot character should be tokenized as a single atom when used as a full stop. That is its most common use. When used as an abbreviation, it should join the text on its left, with white space on its right. But when used in numbers (as “2.2”), uncommon smileys (e.g. “o.o'” or “u.u”), or long URLs (as opposed to shortened URLs), in situations like domains (e.g. “www.seguindo.net”) or in file names (e.g. “.html”), it should remain intact. This character is also involved in some keyboard use errors where a person types “.” when they should be typing “-” — a problem that appears to be more frequent than we expected. It can be said that considering the number of occurrences of the character, it is not the most problematic.

The hyphen should be kept next to its surrounding text when used to create compound words. Some examples from our corpus are “Mercedes-Benz”, “co-infection”, “Port-au-Prince”, “Spider-Man”, “PL-SQL” or “C-130”. Hyphenated compound words are more common in Portuguese, which partially justifies the high number of *False Positives*. The hyphen is also found in a number of popular smileys, symbols regarding arrows, (e.g. “$- >$) and in the middle of some long URLs. It should be isolated when used as a range or dash – for example, when used as a sentence separator in Twitter, standing between the “title” part of the message, and the “message body”; or between the “message body” and a URL at the end. Finally, it is also frequent to use “-” in sport related subjects, to indicate the participants of the match or the resulting score.

The apostrophe is used as an indicator of an unpronounced syllable, as an aggregator of words, as a plural marker for initialisms, and as a possessive indicator for English. In addition, they can also be used as quotation marks (in single or double fashion), or accent replacement as mentioned in Section 3.4.1. Finally, like all other symbols, they find their way into some emoticons. Not all of this situations should be handled in the same way, particularly given the differences in its Portuguese and English semantic meaning.

As the character “:” is very popular in short smileys, almost all of its occurrences in the errors table are related with emoticons.

The *False Positive* errors associated with the character “?” are almost all related with long URLs, while a significant part of the *False Negatives* are related with the white space removal for test purposes, just like the “.”

Table 3.5: The most problematic characters for both scenarios when processing 2500 messages, the errors found, total errors and the number of messages containing the character.

| | $S_{all}$ | | $S_{one}$ | | Total | Messages w/ |
|---|---|---|---|---|---|---|
| Char | FP | FN | FP | FN | errors | occurrences |
| . | 160 | 96 | 812 | 53 | 1121 | 1587 |
| - | 123 | 77 | 586 | 43 | 829 | 477 |
| ′ | 55 | 7 | 342 | 1 | 405 | 86 |
| : | 52 | 19 | 255 | 24 | 350 | 1101 |
| / | 11 | 62 | 40 | 59 | 172 | 725 |
| ? | 22 | 24 | 101 | 18 | 165 | 445 |
| ( | 12 | 6 | 79 | 9 | 106 | 172 |
| ) | 8 | 25 | 43 | 24 | 100 | 327 |

character.

As a *False Negative*, if the user types ". . .", and the test removes some or all the white space, it will not be restored.

The errors produced by the best runs of the SVM classifier for $S_{one}$ were manually analyzed. We present the most common cases.

The character '_' is associated with many False Negatives, and reveals a hard problem to solve. Some usernames end in this character (for example, *"@foo_"*), while others use it to join two words (like *"@foo_bar"*). When the space character after the first username is removed (turning *"@foo_ baz"* into *"@foo_baz"* for the sake of the test) the classifier might accept it as common form of username present in the examples (like *"@foo_bar"* before). , message *"@firstnamelastname_ it would really come in handy xD oh well .."*, it would be illogical to expect it to be restored. Another common error is with the ':' in *smileys*, that produces False Positives, mostly when preceding letters or numbers, as in ":x" or ":O". We expect to address these problems by adding examples that contain more similar cases to the training corpus.

The third character associated with many errors is the '-'. As stated in Section 3.4.1, the '-' symbol is ambiguous, as it can be used to create compound words (should not split, e.g. "middle-aged" or "Coca-Cola"), as well as punctuation (should split). This problem can be solved either by expanding the training set with more relevant examples, or by introducing additional knowledge in the feature generation process. For this purpose, a list of hyphenated compound words can be collected from *Wikipedia* (by mining the titles of articles using DBPedia[6]). This would not compromise the language independence of the system since this resource is available for

[6] http://dbpedia.org/

Table 3.6: Improvement capability of the classifier-based tokenizer.

| Training set size | Testing set size | Incorrect messages | Error rate |
|---|---|---|---|
| 2000 × 5 | 500 × 5 | 511 | 0.204 |
| 2500 | 500 | 110 | 0.220 |
| 2900 | 500 | 89 | 0.178 |

several languages.

### 3.7.1 Augmenting training data based on errors

Table 3.6 presents the values obtained.

All errors that we detected seem solvable by adding more and better examples. To calculate the cost of improving our system's results, we performed an additional experiment. We created a new test set by selecting 500 random messages from our 6 million Twitter messages collection. Using a model trained with all 2500 messages in our previous corpus, we noticed 110 of the test messages were incorrectly tokenized. We then augmented our corpus with 400 new messages — 50 new messages containing each of the eight problematic characters in Table 3.5. The new model, trained with the now 2900 messages corpus, failed in only 89 messages out of the previous 500. That is a reduction of almost 20% in the number of messages tokenized incorrectly.

With little more than 2 hours of work from a non-expert (as the only requirement is for them to understand and follow the tokenization rules stated in Section 3.4), the classification system improved by a significant amount. Manual tokenization of examples is a highly paralleliazable activity. By comparison, rule creation is a very repetitive process of adding or correcting a rule and then testing it for effectiveness or interference with the others. This possibility of interference also limits the process to one single person that also needs to be an expert in the relevant pattern matching mechanism used (e.g. regular expressions).

## 3.8 Conclusion and future work

We showed that the problem of tokenization of UGC, although fundamental for subsequent language processing work, cannot be considered a simple task. The usual UGC (and microblogging in particular) way of expression can be problematic for "traditional" methods of text processing. In tokenization this is particularly evident. The difficulty arises from the lack of editorial rules and few accepted standards, meaning that ambiguity is not always avoidable. In particular, no character (other than white space) can be said to always mark the start or end of a token — or so we thought, until we saw it being used as a method of obfuscation (see Table B.1 on page 187 for a list of other exceptions).

We created a corpus of Twitter messages that were tokenized manually following our rules for UGC tokenization. We then compared the results obtained by a classification-based and a rule-based tokenizer across two scenarios, that draw a typical upper and lower limit of performance for most microblogging usage.

In our experiments, we achieve F1 values of 0.96 for scenario $S_{all}$ (that tests Recall) and 0.88 for scenario $S_{one}$ (that focuses in Precision), for our classification-based tokenizer, when trained with 2000 examples. Our results can be said in-line with comparable systems in the literature.

We have also shown that text classification methods can successfully tackle this problem, and significantly better than rule-based classifiers. Not only do they achieve better performance (based on F1 values) but also their performance can be improved significantly by simply adding to the training set more examples regarding the problematic situations. The text classification approach we propose is also better in terms of development costs, since the bulk of the work (i. e., the annotation) can be distributed among a group of non-experts.

Having said that, there is still a long road to go before tokenization can be considered a solved problem. We have a number of ideas noted down that we would like to try.

For example, we intend to experiment with the SVM kernels and parameters, looking for an optimal classification configuration. We also wish to compare the results of the SVM with other classification algorithms, such as Conditional Random Fields (CRF). In addition, we also consider extending our feature set to include, for instance, information about keyboard layouts (e.g. which keys are next to the one that produces this character) as well as features not based on a singular character, such as determining if the relevant *decision point* is within a URL, recognized using regular expressions.

Looking even further ahead, producing language and/or user specific models could result in even better results, as more specific patterns (such as preferences for certain emoticons) can be identified.

Stepping aside from what could be considered "incremental improvements", we would like to experiment with a multi-stage tokenization process. This would be language-dependent, working on three levels through a backtracking process in search of the better solution (from the most obvious to the most obscure possibilities). The three stages would be:

**bite** Propose a position for the end for the token; start removing spaces if necessary.

**chew** Look at the proposed token and look it up on knowledge bases (is it a dictionary word, a proper name, a new trend word, or perhaps a foreign word?). How likely is this an obfuscated word if all else fails?

**swallow** How well does the sentence fit together (grammatically, semantically…). Score it. We will take the best score.

# Chapter 4

# Forensic analysis

**Contents**

In this chapter we propose a set of stylistic markers for automatically attributing authorship to micro-blogging messages. The proposed markers include highly personal and idiosyncratic editing options, such as ‘emoticons’, interjections, punctuation, abbreviations and other low-level features.

We evaluate the ability of these features to help discriminate the authorship of Twitter messages among three authors. For that purpose, we train SVM classifiers to learn stylometric models for each author based on different combinations of the groups of stylistic features that we propose.

Results show a relatively good-performance in attributing authorship of micro-blogging messages ($F1 = 0.63$) using this set of features, even when training the classifiers with as few as 60 examples from each author ($F1 = 0.54$). Additionally, we conclude that emoticons are the most discriminating features in these groups.

## 4.1 Introduction

In January 2010 the *New York Daily News* reported that a series of Twitter messages exchanged between two childhood friends led to one murdering the other. The set of Twitter messages exchanged between the victim and the accused was considered a potential key evidence in trial, but such evidence can be challenged if and when the alleged author *refutes* its authorship. *Authorship analysis* can, in this context, contribute to confirming or excluding the hypothesis that a given person is the true author of a queried message, *among several candidates*. However, the micro-blogging environment raises new, significant challenges as the messages are *extremely short* and fragmentary. For example, Twitter messages are limited to 140 characters, but very frequently have only 10 or even fewer words. Standard stylistic markers such as *lexical richness*, *frequency of function words*, or *syntactic measures* — which are known to perform well with longer, 'standard' language texts — perform worse with such short texts, whose language is 'fragmentary' [Gra10]. Traditional authorship analysis methods are considered unreliable for text excerpts smaller than 250-500 words, as the accuracy tends to drop significantly with text length decrease [HF07].

In this chapter we use a text classification approach to help us investigate whether some 'non-traditional' stylistic markers, such as the type of emoticons, provide enough stylistic information to be used in authorship attribution. Given a set of Twitter messages from a group of *three* authors and sets of *content-agnostic* stylistic features, we train classifiers to learn the stylometric models for each author. We test whether the classifiers can robustly and correctly attribute authorship of an unseen message written by one of the three authors using the proposed features. We focus specifically on Twitter for its popularity, and address Portuguese in particular, which is one of the most widely used languages in this medium[1].

## 4.2 Related work

In recent years, there has been considerable research on authorship attribution of some *user-generated contents* — such as *e-mail* (e.g. [dVACM01]) and, more recently, *web logs* (e.g. [PLZC09, GSR09, KSA09]) and 'opinion spam' (e.g. [JL08]). However, to the best of our knowledge, research on authorship attribution of Twitter messages has been scarce, and raised robustness problems. And before our work, it was non-existent in Portuguese. Research on authorship attribution in general in Portuguese has been very limited, and has focused mainly on online newspapers data (e. g. [PJO07, SSSG+10]).

[1] `http://semiocast.com/downloads/Semiocast_Half_of_messages_on_Twitter_are_not_in_English_20100224.pdf`

The biggest challenge in attributing authorship of micro-blogging messages is their *extremely short* nature. In general, studies on intrinsically short and fragmentary messages focus on other interactive dimensions of the media, such as analysing the dialogue structure of Twitter conversations [RCD10], detecting trends and tracking memes [CL09, Che] on the Twittosphere, or performing automatic question-answering through text messaging (e. g. [KNF+09]).

To tackle the problem of robustness in computational stylometric analysis, research (e.g. the 'Writeprints technique' [AC08]) was applied to four different text genres to discriminate authorship and detect similarity of online texts among 100 authors. The performance obtained was good, but (a) the procedure did not prove to be content-agnostic, and (b) did not analyse Twitter messages. Also, using structural features that are possibly due to editing and considering 'idiosyncratic features' usage anomalies to include misspellings and grammar mistakes, and leaving personal choice partly aside, it is bound to compromise the results.

One of the first tasks to authorship attribution on Twitter consisted of detecting spammers. Benevenuto et al [BMRA10] enumerate a set of characteristics to distinguish spamming and non-spamming accounts. The authors address *content* clues (e. g. fraction of tweets with URLs, average number of URLs per tweet, number of tweets the user replied to, and average number of hashtags per tweet) and also *behaviour* clues (e. g. age of the user account, fraction of followers / followees, fraction of tweets replied by user, number of tweets the user received in reply, number of followees and followers). The authors used an SVM classifier to correctly identify 70% of the spammers and 96% of non-spammers using a 5-fold cross-validation process, applied to a corpus of 1,065 users containing 355 spammers manually identified. The authors concluded that both features produce similarly useful clues, but that the *behaviour* features were harder to mask.

More recently, it has been demonstrated that the authorship of twitter messages can be attributed with a certain degree of certainty [LWD10]. Surprisingly, the authors concluded that authorship could be identified at 120 tweets per user, and that more messages would not improve accuracy significantly. However, their method compromises the authorship identification task of most unknown messages, as they reported a loss of 27% accuracy when information about the interlocutor's user data was removed.

It has also been demonstrated that authorship could be attributed using 'probabilistic context-free grammars' [RKM10] by building complete models of each author's (3 to 6) syntax. Nevertheless, the authors used both syntactic and lexical information to determine each author's writing style. Even other authors [BFD+10] who focused on short classification for information filtering, rather than authorship attribution, used domain-specific features, lexical items and participant information. Both *content* and *behaviour* features produce similarly useful clues, as determined by the $\chi^2$

rankings, but of the 62 total features used, the 10 most important were:

1. The fraction of tweets with URLs,
2. the age of the user account,
3. the average number of URLs per tweet,
4. the fraction of followers per followees,
5. the fraction of tweets the user had replied,
6. the number of tweets the user replied,
7. the number of tweets that originated a reply,
8. the number of followees,
9. the number of followers, and
10. the average number of hashtags per tweet.

Conversely, we propose a content-agnostic method, based on low-level features to identify authorship of unknown messages. This method is independent of user information, so not knowing the communication participants is irrelevant to the identification task. Moreover, although some of the features used have been studied independently, this method is innovative in that the specific combination of the different stylistic features has never been used before and has not been applied to such short texts.

## 4.3 Method description & stylistic features

Authorship attribution can be seen as a typical *text classification* task: given examples of messages written by a set of authors (classes), we aim to attribute authorship of messages of unknown authorship. In a forensic scenario, the task consists of discriminating the authorship of messages of a small number of potential authors (e. g. 2 to 5), or determining whether a message can be attributed to a certain ('suspect') author.

The key to framing authorship attribution as a text classification problem is the selection of the feature sets that best describe the *style* of the authors. We propose four groups of stylistic features for automatic authorship analysis, each dealing with a particular aspect of tweets. All features are *content-agnostic*; to ensure a robust authorship attribution and prevent the analysis from relying on topic-related clues, they do not contain lexical information.

### 4.3.1 Group 1: Quantitative Markers

These features attempt to grasp simple quantitative style markers from the message as a whole. The set includes message statistics, e. g. length (in characters) and number of tokens, as well as token-related statistics (e. g. average length, number of 1-character tokens, 2-consonant tokens, numeral tokens, choice of case, etc). We also consider other markers, e. g. use of dates, and words not found in the dictionary[2] to indicate possible spelling mistakes or potential use of specialised language.

As Twitter-specific features, we compute the number of user references (e. g. @user_123), number and position of *hashtags* (e. g. #music), in-message URLs and the URL shortening service used. We also take note of messages starting with a username (a reply), as the author may alter their writing style when addressing another person.

### 4.3.2 Group 2: Marks of Emotion

Another highly personal — and hence idiosyncratic stylistic marker — is the device used to convey emotion. There are mainly three non-verbal[3] ways of expressing emotion in user-generated contents:

- *smileys*;
- *‘LOLs’*; and
- *interjections*.

*Smileys* (‘:-)’) are used creatively to reflect human emotions by changing the combination of eyes, nose and mouth. This work explores three axes of idiosyncratic variation: *range* (e. g. number of happy smileys per message), *structure* (e. g. whether the smiley has a nose) and *direction* of the smiley. First, different users express different *ranges* of emotion (for example, sadness ‘:-(’, happiness ‘:-D’, worry ‘:-S’, playfulness ‘:-P’, surprise ‘:-O’ or frustration ‘X-[’). Second, users structure the smiley differently; some authors prefer the ‘noseless’ look (‘:)’), while others emphasise the mouth (e. g. ‘:-))’); likewise, the eyes of the *smiley* are also subject to variation: besides their most common format (i.e. ‘:’), they can also be winking (‘;’) or closed (‘|’, ‘X’). Third, different users direct smileys differently; some users choose to use them *right-facing* (e. g. ‘:-)’), while others prefer them *left-facing* (e. g. ‘(-:’); others use the much less common *upright* direction, e.g.: happy ‘^_^’, very happy ‘^___^’, frustrated ‘>_<’ or crying ‘T_T’).

Another form of expression is the prevalent ‘LOL’, which usually stands for *Laughing Out Loud*. Frequently users manipulate the basic ‘LOL’ and

[2] We use the GNU Aspell dictionary for European Portuguese.

[3] In Twitter, users can also add hashtags to the message to signal a certain emotion or mood (e. g. "#sarcastic"), but we only focus on the three more general devices.

'maximise' it in various other forms, e. g. by repeating its letters (e. g. 'LLOOOLLL') or creating a loop (e. g. 'LOLOL'). The 'LOL' acronym is so prevalent in Internet slang that it is frequently typed in lowercase and used as a common word (e. g. 'I lolled'). This subgroup describes several instances of *length*, *case* and *ratio* between 'L' / 'O', so as to distinguish between 'LOL' and the exaggeration in multiplying the 'O', as in 'LOOOOL'.

We identify interjections as tokens consisting of only two alternating letters that are not a 'LOL', such as 'haaahahahah'. Other popular and characteristic examples are the typical Brazilian laughing 'rsrsrs' and the Spanish laughing 'jejeje' — both of which are now commonly found in European Portuguese Twitter. We count the number of interjections used in a message, their average length and number of characters.

### 4.3.3 Group 3: Punctuation

The choice of punctuation is a case of writing style [Eag94], mostly in languages whose syntax and morphology is highly flexible (such as Portuguese and Spanish). Some authors occasionally make use of expressive and non-standard punctuation, either by repeating ('!!!') or combining it ('?!?'). Others simply skip punctuation, assuming the meaning of the message will not be affected. Ellipsis in particular can be constructed in less usual ways (e. g. '..' or '......'). We count the frequency of these and other peculiar cases, such as the use of punctuation after a 'LOL' and at the end of a message (while ignoring URLs and *hashtags*).

### 4.3.4 Group 4: Abbreviations

Some abbreviations are highly idiolectal, thus depending on personal choice. We monitor the use of three types of abbreviations: 2-consonant tokens (e. g. 'bk' for 'back'), 1- or 2-letter tokens followed by '.' or '/' (e. g. 'p/') and 3-letter tokens ending in two consonants, with (possibly) a dot at the end (e. g. 'etc.').

## 4.4 Experimental setup

This study is focused on the authorship identification of a message among three candidate authors. We consider only three possible authors as forensic linguistic scenarios usually imply a limited number of suspect authors, and is hence more realistic (but our approach has been tested successfully with as many as *ten* candidate authors). We chose to use Support Vector Machines (SVM) [Joa98] as the classification algorithm for its proven effectiveness in text classification tasks and robustness in handling a large number of features. The *SVM-Light* implementation [Joa98] has been used, parametrised to a linear kernel. We employ a *1-vs-all* classification strategy; for each author, we use a SVM to learn the corresponding stylistic model, capable of discriminating each author's messages. Given a suspect message from each author, we use each SVM to predict the degree of likelihood that each author is the true author. The message authorship is attributed to the author of the highest scoring SVM. We also consider a threshold on the minimum value of the SVM score, so as to introduce a *confidence* parameter (the minimum score of the SVM classifier considered valid) in the authorship attribution process. When none of the SVM scores achieves the minimum value, authorship is left undefined.

Our data set consists of Twitter messages from authors in Portugal, collected in 2010 (January 12 to October 1). We counted over 200,000 users and over 4 million messages during this period (excluding messages posted automatically, such as news feeds). From these, we selected the 120 most prolific Twitter authors in the set, responsible for at least 2,000 *distinct* and *original* messages (i.e. excluding *retweets*), to extract the sets of messages for our experiments. The messages were all tokenized using a UGC-specific method that takes into account its typical writing style, Internet slang, URLs and Twitter usernames and *hashtags*. We divide the 120 authors into 40 groups of 3 users at random, and maintain these groups throughout our experiments. The group of 3 authors forms the basic testing unit of our experiment. Each test message from the group's data set is attributed to one of the 3 authors using the pre-calculated stylistic models, or to none if the minimum confidence threshold is not reached.

We perform two sets of experiments. In *Experimental Set 1*, the classification procedure uses all possible groups of features to describe the messages. We use data sets of sizes 75, 250, 1,250 and 2,000 messages/author. In *Experimental Set 2*, we run the training and classification procedure using *only one* group of features at a time. We use the largest data set from the previous experiment (2,000 messages/author) for this analysis.

### 4.4.1 Performance evaluation

We measure *Precision* ($P$), *Recall* ($R$) and $F1$ ($2PR/(P+R)$) considering:

$$P = \frac{\text{\# messages correctly attributed}}{\text{\# messages attributed}}$$

$$R = \frac{\text{\# messages correctly attributed}}{\text{\# messages in the set}}$$

We run the training and classification procedures in each set of experiments and use the *confidence* parameter to draw *Precision vs. Recall graphs.* As these experiments consider *three* different authors, the baseline is $F1 =$

Table 4.1: Best $F1$ values obtained for different data set sizes.

| Data set size | 75 | 125 | 250 | 625 | 1,250 | 2,000 |
|---|---|---|---|---|---|---|
| $F1$ | 0.54 | 0.55 | 0.57 | 0.59 | 0.61 | 0.63 |

0.33 ($P = 0.33$ at $R = 0.33$). All experiments were conducted using a 5-fold cross validation, and run for all 40 groups of 3 authors. For varying levels of Recall (increments of 0.01) we calculate the maximum, minimum and average Precision that was obtained for all 40 groups. All $F1$ values are calculated using the average Precision.

## 4.5 Results and analysis

### 4.5.1 Experiment set 1

Figure 4.1 shows the Precision vs Recall graphs for Experimental Set 1. Data set increases (from 75 to 2,000 messages/author) returns improvements in the minimum, maximum and average Precision values. In addition, the robustness of the classifier also benefits from the added examples, as the most problematic situations (corresponding to the minimum precision values) are handled correctly more frequently. The best $F1$ values are always obtained at the highest value of Recall, meaning that they too follow this improvement trend.

For the smaller data set (75 messages, Figure 4.1a), the minimum Precision curve is nearly constant, not showing a benefit from the decision threshold (at the cost of Recall). We speculate this is due to two reasons. First, given the large feature space (we use at least 5680 dimentions), and the relatively small number of non-negative feature component in each training example (most messages have between 64 and 70 features), a robust classification model can only be inferred using a larger training. Second, with such small sets it is highly probable that both the training and the test sets are atypical and distinct in terms of feature distribution. Still, the performance values obtained are far above the baseline, and an $F1$ value of 0.54 is reached. In the larger data sets (Figures 4.1c and 4.1d) we always obtain a Precision greater than 0.5. This means that even in the more difficult cases, the attribution process is correct more often than not. However, the contribution of the extra examples for the $F1$ values is lower when we exceed 250 messages/author (where we get 0.59), even if they increase almost linearly up to 0.63 (for 2,000 messages/author).

These observations are relevant since most previous work argued that longer strings of text were necessary to attribute authorship. Better results can be obtained if more than one message is available, and we know them to be of the same author. In this situation, the probability of making a mistake in authorship attribution drops exponentially (following a binomial distribution for repeated trials).

### 4.5.2 Experiment set 2

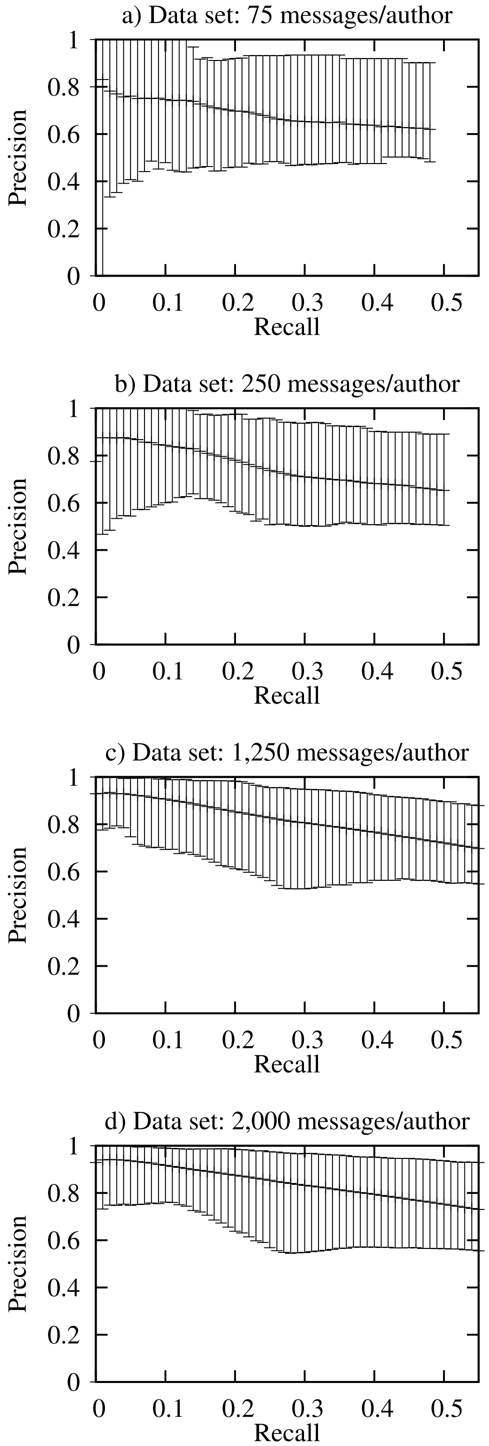
Figure 4.2 presents the Performance vs Recall curves for authorship attribution with a classification procedure using *only one* group of features at a time. Quantitative Markers (Group 1) show an average performance, with minimum Precision and maximum Recall of 0.49, and maximum $F1$ value of 0.55 (Figure 4.2a). This shows that, albeit Twitter length constraints, there is room for stylistic choices like length of tokens, length of message posted, etc. Markers of *expression of emotion*, including *smileys*, *LOLs* and *interjections* (Group 2) achieve a relatively high performance, and clearly outperform all other feature groups (Figure 4.2b). It achieves an $F1$ value of 0.62 (where using *all* features together achieves 0.63). This is particularly interesting since these features are specific to user-generated contents, and to our knowledge their relevance and effectiveness in authorship attribution is now quantified for the first time. The difference between the average and minimum Precision values is an indicator that the low performance of this feature group is an infrequent event. The group of features including punctuation (Group 3) performs slightly worse than the previous groups, and scores only 0.50 on the $F1$ measure (Figure 4.2c). The difference between the best and worse case is significant, but the average Precision degrades as the Recall increases. Our evaluation demonstrates that our approach, although quite simplistic, is capable of detecting stylistic variation in the use of punctuation, and of successfully using this information for authorship attribution. This result is in line with those reported previously by [SSSG$^+$10] for punctuation-based features applied to automatic authorship attribution of sentences from newspapers. Group 4, containing features on the use of abbreviation, led to the worst results (maximum $F1$ value of 0.40). The shape of the curve rapidly approaches the baseline values, proving that this group is not robust (Figure 4.2d). Manual evaluation shows that these abbreviations are used rarely. However, as the low Recall/high Precision part of the curve suggests, they carry stylistic value, in spite of being used only in a relatively small number of cases. Finally, the performance when using *all* groups of features simultaneously (Figure 4.1f) is better than using any group of features *individually*, showing that all individual groups of features carry relevant stylistic information that can be combined, and suggesting that the investment in devising new groups of stylistic features may lead to additional global performance improvements — especially the recall. It is apparent in Figure 4.2 that the last graph achieves larger values of Recall. This can be explained by the absence of some features from a number of messages, or that those that were found were, on their own, incapable of providing results (e. g. the message size).

## 4.6 Conclusions

Our experiment demonstrates that standard text classification techniques can be used in conjunction with a group of content-agnostic features to successfully attribute authorship of Twitter messages to three different authors. Automatic authorship attribution of such short text strings, using only *content-agnostic* stylistic features, had not been addressed before. Our classification approach requires a relatively small amount of training data (as little as 100 example messages) to achieve good performance in discriminating authorship.

These results owe to the fact that the attribution is based on a good and robust set of features that reflect the stylistic choices of the authors, while still being content-agnostic.

Surprisingly, the group of emoticons outperforms each of the other feature groups tested, with a relatively high performance. The relevance and effectiveness of these features for automatic authorship attribution are now demonstrated for the first time. Features related to use of abbreviation report the poorest results, their performance dropping abruptly; yet, despite lacking robustness, they prove to carry stylistic information. The group of punctuation marks is the second best performer, showing its capacity to capture stylistic variation. Quantitative and punctuation markers show average results, carrying some idiolectal information, despite the text length constraints. On balance, it can be argued that all features carry relevant information, since using all groups of features simultaneously allows inferring more robust authorship classifiers than using any group of features individually.



a) Data set: 75 messages/author

b) Data set: 250 messages/author

c) Data set: 1,250 messages/author

d) Data set: 2,000 messages/author

Figure 4.1: Performance of each data set size. Each graph plots maximum, average and minimum Precision at varying levels of Recall (40 groups of 3 authors).

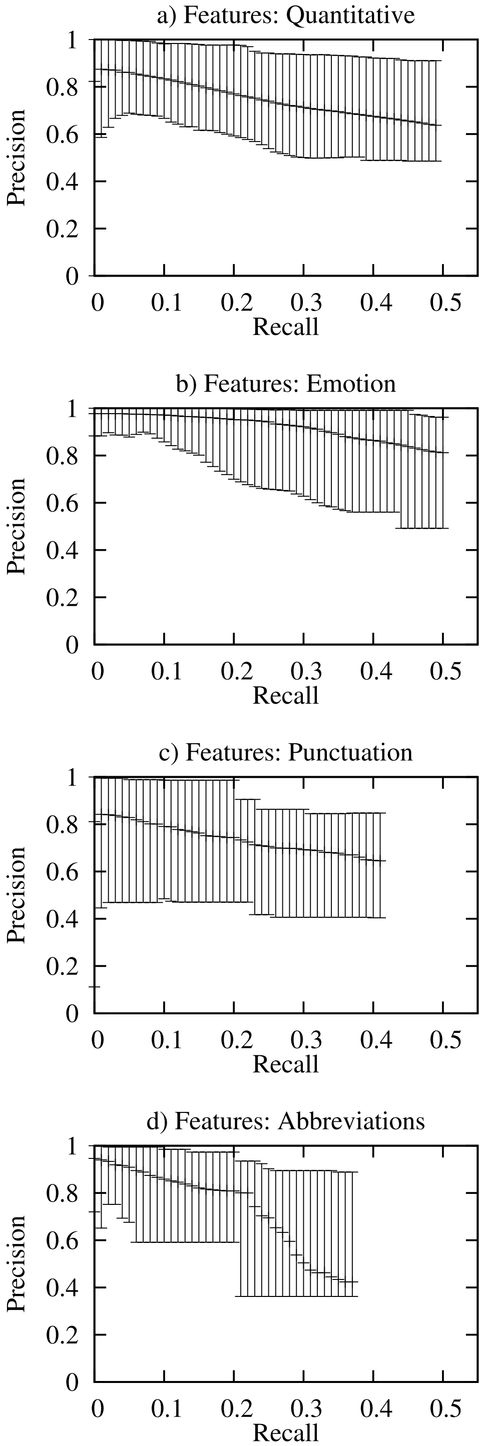


Figure 4.2: Performance of each individual set of features. Each graph plots maximum, average and minimum Precision at varying levels of Recall (40 groups of 3 authors, 2,000 messages from each author).

# Chapter 5

# Bot detection

## Contents

In this chapter we study the problem of identifying systems that automatically inject non-personal messages in micro-blogging message streams, thus potentially biasing results of certain information extraction procedures, such as opinion-mining and trend analysis. We also study several classes of features, namely features based on the time of posting, the client used to post, the presence of links, the user interaction and the writing style. This last class of features, that we introduce here for the first time, is proved to be a top performer, achieving accuracy near the 90%, on par with the best features previously used for this task.

## 5.1 Introduction

Microblogging systems — of which Twitter is probably the best known example — have become a new and relevant medium for sharing spontaneous and personal information. Many studies and applications consider microblogs as a source of data, precisely because these characteristics can confer authenticity to results. For example, *trend detection* ( [MK10]), *opinion-mining* ( [DH09]) or *recommendation* ( [CNN$^+$10]).

Because of its popularity, Twitter is also part of the on-line communication strategy of many organizations, which use a Twitter account for providing updates on news, initiatives, commercial information (e.g. promotions, advertisements and spam) and various other types of information people may find interesting (like weather, traffic, TV programming guides or events).

Messages conveyed by these automatized accounts – which we will now refer to as *robot accounts* or, simply, *bots* – can easily become part of the stream of messages processed by information extraction applications. Since bots provide content aimed at being consumed by the masses instead of the personal messages that information extraction systems consider meaningful (for example, for trend detection), automatic messages may bias the results that some information extraction systems try to generate. For this reason, from the point of view of these systems, messages sent by bots can be considered noise.

The number of such robot accounts is extremely large and is constantly growing. Therefore, it is practically impossible to manually create and maintain a list of such accounts.

Even considering that the number of messages typically produced by a bot each day is not significantly larger than the number of messages written by an active user in the same period, we must remember that bots are capable of sustaining their publication frequency for longer periods than most humans (that can stop using the service temporarely or permanently after some time). Thus, in the long run, bots are capable of producing a larger set of messages than an active person.

In this work we propose a system that can identify these robot accounts using a classification approach based on a number of observable features related to activity patterns and message style. This system cannot detect *every form of bot*, but it is focused on detecting a number of characteristics that are common on many automated accounts (possibly in the majority). We evaluate its performance, and compare it with some of the more common approaches used for this task, such as the client used to post the messages and the regularity of new content.

### 5.1.1 Types of users

Based on the work of Chu et al. [CGWJ10], we start by distinguishing between three types of users. The term *human* is used to refer to users that author all or nearly all their messages. They usually interact with other users, post links on some of the messages, use abbreviations, emoticons and ocasionally misspell words. Many employ irregular writing styles. The subject of their messages can be different, but they tend to express personal opinions. Below we have examples of two human users:

- Who's idea was it to take shots of tequila? You are in so much trouble.
- I forgot to mention that I dropped said TV on my finger. ouchie.
- Heard that broseph. RT @ReggaeOCD: So bored with nothing to do. #IHateNotHavingFriends
- Just being a bum today. `http://twitpic.com/4y5ftu`
- aw, grantly:'destroys only happy moment in fat kids life' when talking about food :@
- @ryrae HAHAHAHAHHAHA :')
- JOSH IS IN SEASON 5 OF WATERLOO ROAD! WHEN DID THIS HAPPEN?

*Bots*, on the other hand, are in place to automate the propagation of information. The content is generally written by a person, although in some cases the entire process is automated (e.g. sensor readings).

We should note that what we are distinguishing here is more a matter of content than a matter of process or form. It is possible that an account where a person writes the message directly at the Twitter website be labeled as a bot, if the messages are written in the cold objective way seen in the examples below, from three different accounts.

- Social Security and Medicare to run short sooner than expected. `http://on.cnn.com/lauSNv`
- For Louisiana town, a collective gasp as it braces for floodwaters. `http://on.cnn.com/mb481c`
- Jindal: Morganza Spillway could open within the next 24 hours. `http://on.cnn.com/j2jIBs`
- 96kg-Bosak takes 7th place at the University Nationals

- 84kg-Lewnes takes 2nd to Wright of PSU and qualifies for the world team trials in Oklahoma Ciry
- Bosak loses his consolation match 0-1, 1-3 to Zac Thomusseit of Pitt.
- #Senate McConnell: Debt limit a 'great opportunity' `http://bit.ly/kanlc9` #Politics
- #Senate Wisconsin Sen. Kohl to retire `http://bit.ly/kQpAsS` #Politics
- #Senate Ensign may face more legal problems `http://bit.ly/mN7XsB` #Politics

Many accounts are not run entirely by a person nor are they completely controlled by a machine. We label these mixed accounts as *cyborgs*, the term used by Chu et al. [CGWJ10], that describes them as a "bot-assisted human or human-assisted bot". For example, an enterprise can have an automatic posting service, and periodically a person provides the user interaction to maintain a warmer relation with the followers, and foster a sense of community. Another possibility occurs when a person uses links to websites that post a message on the account of that person (for example, "share this" links). If these pre-written content are noticeable among the original messages, the user is labeled as a cyborg. If barely no original content is present in the user's timeline, they will be considered a bot. Below we show examples of a cyborg account.

- Explore The Space Shuttle Era `http://go.nasa.gov/gzxst5` and immerse yourself in the Space Shuttle Experience `http://go.nasa.gov/iHVfGN`
- Track the space shuttle during launch and landing in Google Earth using real-time data from Mission Control `http://go.nasa.gov/mwO9Ur`
- RT @Rep_Giffords: Gabrielle landed safely @NASAKennedy. For more details go to `www.fb.me/GabrielleGiffords`. #NASATweetUp
- Space shuttle Endeavour's preferred launch time moved two seconds later! Now 8:56:28 a.m. EDT Monday.
- @Angel_head NASA frequently tweaks the shuttle launch time by seconds based on the latest space station tracking data to use the least fuel.

As we will explain next, we used these guidelines to construct a Ground Truth that will be used in our experiments. The details of this task are given in Section 5.4.1.

To address the problem of automatic posting, we study different sets of features that allow us to classify Twitter users into the three user categories described. These features explore characteristics exhibited by the users, such as their posting times, the microblog client application they use, their interaction with other users, the content of their messages, and their writing style. The main goal of the work presented in this chapter is to access the usefulness and robustness of the different types of features proposed. We discuss the features in Section 5.3.

We describe our experiment and its parameters in Section 5.4 and evaluate our results in Section 5.5. In Section 5.6 we present our conclusions and future work.

## 5.2 Related work

Most literature addressing the identification of automated systems in microblogs is related with the detection of spam. While there is some overlapping between spam and automated posting systems (spammers often employ automation to help them in their work), we feel that the problem we are addressing is much more general.

Wang [Wan10] presents an effort to detect spamming bots in Twitter using a classification based approach. The author explored two sets of features: (i) information about the number of followers, friends and the follower per friend ratio for the social network aspect of Twitter; and (ii) information about duplicate content and number of links present in the last 20 messages of a given user account. The author used a manually annotated corpus to train a Naive-Bayes classifier. The classifier achieved slightly over 90% Precision, Recall and F-measure in a 10-fold cross validation experiment. However, since the training corpus was biased towards non-spam users (97% of the examples), any classifier that only reported "non-spam" would be almost always correct, so results are not really significant.

Grier et al. [GTPZ10] analyze several features that indicate spamming on Twitter. They looked for automated behavior by inspecting the precision of timing events (minute-of-the-hour and second-of-the-minute), and the repetition of text in the messages across a user's history. They also studied the Twitter client application used to write the messages, since some allow to pre-schedule tweets at specific intervals.

Zhang and Paxson [ZP11] present a study where they try to identify bots by looking only at the times, more specifically, the minute-of-the-hour and the second-of-the-minute. If the posting times are either too uniform or not uniform enough, there is the possibility of the account being automated. This analysis is similar to the one present in Grier et al. [GTPZ10], a work where Zhang and Paxson participated.

The authors present no validation of their results (since it is not possible to determine for sure if the account is automated or not). However, they claim that "11% of accounts that appear to publish exclusively through the browser are in fact automated accounts that spoof the source of the updates".

Chu et al. [CGWJ10] propose to distinguish between humans, bots or cyborgs, but much of the effort was put into spam detection. They claim that more sophisticated bots unfollow users that do not follow back, in an effort to keep their friends to followers ratio close to 1, thus reducing the effectiveness of features based on the social network of Twitter. The official validation of accounts is rare, and the date of their creation are also said to be less useful. Despite its interesting observations, the article has a number of problems that we feel should be pointed out.

There are too many places in the article where Chu et al. seem to be mistaking Twitter with a website (e.g. what is an "external link" in a Twitter message?). We also have some reservations about their data, as it defies common observation and our measurements (e.g. Twitter users being the most active at 6 AM local time, and not posting at midnight), but it could be explained by cultural differences.

Finally, the methods employed by Chu et al. are not explained clearly, and can be questioned. For example, they state that "Training the component with up-to-date spam text patterns on Twitter helps improve the accuracy", but fail to mention how and where they collect such information. They also omit the size of their spam and non-spam dataset, stating only that it "consists of spam tweets and spam external URLs, which are detected during the creating of the ground truth set."

The most serious methodological problem occurs during the compilation of their ground truth sets, where Chu et al. state that "The samples that are difficult and uncertain to classify fall If the ambiguous and difficult to classify examples are excluded from the experiment, we can seriously question the generality of their results.

Contrary to previous work, we focus on a problem that is much more generic than spam-detection, since a very large number of bots belong to newspapers or other organizations that are voluntarily followed by users. Our goal is to separate potentially opinionated and highly personal content from content injected in the Twittosphere by media organizations (mostly informational or promotional). One other point where we distinguish ourselves from the mentioned works is in our attempt to expand the set of features used in the detection, now including a vast array of stylistic markers. In our opinion, this opens a new field for exploration and study.

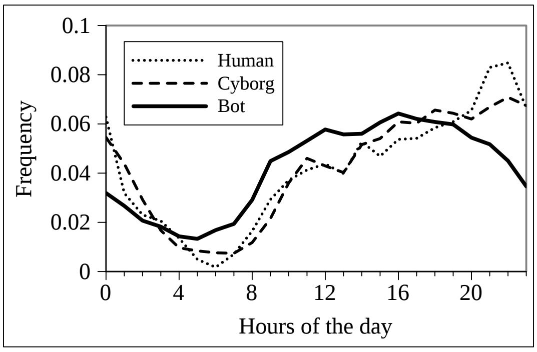


Figure 5.1: Comparison of Twitter activity between bots, cyborgs and humans as a function of the hour.

## 5.3 Methodology

Most of our discussion is centered around distinguishing human users and bots. We propose five sets of features, described below, that are intended to help to discriminate between these two poles. A cyborg user, by definition, exhibits characteristics typical of both classes of users.

### 5.3.1 Chronological features

One of the characteristics of automatic message posting systems is that they can be left running indefinitely. Therefore, we can expect to see different chronological patterns between human users and bots. To address these points, we defined a number of features, divided into the four following sub-classes.

#### Resting and active periods

Constant activity throughout the day is an indication that the posting process is automated or that more than one person is using the same account — something we expect to be unusual for individual users. Figure 5.1 was drawn using information from our manually classified users (described in Section 5.4.1). It shows how human and cyborg activity is reduced between 1 and 9 AM. Bot activity is also reduced between 11 PM and 7 AM, but it never approaches zero. We can understand this drop by the absence of an audience to read the content being produced; however, messages posted in the very early morning are more likely to be read when people look at their feed for the first time during the day, which explains the rise of bot posting starting at 5 in the morning.

At the same time, other things can keep people from blogging. This can lead to a certain hour of preferred activity, such as evenings, as suggested

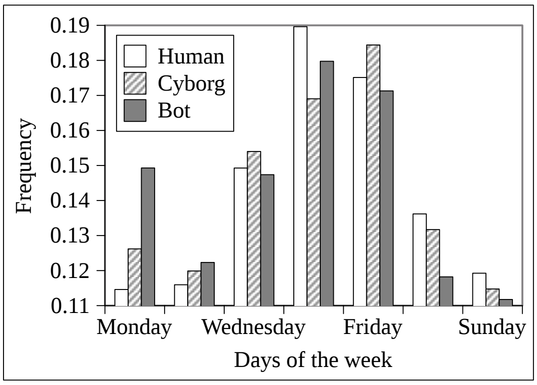


Figure 5.2: Comparison of Twitter activity between bots, cyborgs and humans as a function of the day of the week.

in Figure 5.1. Bots appear to have a more evenly spaced distribution across the day, with smaller fluctuations in the level of activity. This can be a conscious choice, to allow more time for their followers to read each post.

Since we tried to limit our crawling efforts to Portugal, most of the observations are expected to fall within the same time zone (with the exception of the Azores islands, which accounts for 2% of the population). We believe that the problem of users in different time zones cannot be avoided completely. For example, we do not expect users to correct their Twitter profile when traveling.

To detect the times at which the user is more or less active, we define 24 features that measure the fraction of messages they posted at each hour. These values should reflect the distributions represented in Figure 5.1. We also analyze the average and standard deviation of these values. We expect that the standard deviation of a bot is lower than that of a human.

Finally, we register the 10 hours with the highest and lowest activity, and the average number of messages that the user posts per day (as a floating-point number).

### Long term activity

Days are not all equal. This is true for both humans and bots. For example, as shown in Figure 5.2, most activity happens at Thursdays and Fridays. This trend matches the result published by Hubspot [Bur10].

We can see that bots are less active during the weekends, while they dominate on Mondays and lead on Tuesdays.

We define seven features related to the frequency of the messages posted across each day of the week, that should reflect the proportions in Figure 5.2. We also calculate the workday and weekend posting frequency, and rank the days of the week by the frequency of posting.

### Inactivity periods

From direct observation of Twitter messages, we can see that bots tend to be more regular on their updates than humans. It is known that irregular accounts can lose popularity quickly. At the same time, normal people need to rest, get occupied with other matters, and can lose interest in blogging for some time.

To make use of this information, we measure the periods of inactivity in minutes, and record the length of the 10 longest intervals, in decreasing order. We also calculate the average and standard deviation of all these values. From our observations, we expect that bots will have lower variation in inactivity periods (lower standard deviation) and a higher average.

For example, considering a user that only blogged at 1 PM, 2 PM, 3:30 PM and 7 PM on the same day, we would have the following features:

| Feature name | Value (minutes) |
|---|---:|
| top_inactive_period_1 | 210.00 |
| top_inactive_period_2 | 90.00 |
| top_inactive_period_3 | 60.00 |
| average_top_inactive_period | 120.00 |
| standard_deviation_top_ inactive_period | 79.37 |

Humans are unable to match the speed at which bots can create new messages. For this reason, we also calculate the analogous features for the minimum inactivity periods (i.e. the 10 shortest inactive periods).

### Posting precision

Since some automatic posting systems work based on a fixed periodicity (e.g. TV programming guides), we decided to calculate the frequency of messages that are created at each minute (60 features) and second (another 60 features). This approach is a simpler version of other works [GTPZ10, ZP11].

For a human, we expect their posts to be evenly spread across both these measurements. Some bots, on the other hand, are expected to concentrate their activity around the 0 seconds mark. They may also do the same around some particular minutes (e.g. 0 or 30).

As before, we also calculate the average and standard deviation of these measurements, where humans should result in a lower average and higher standard deviation compared to bots.

For both minutes and seconds we take note of the 10 most frequent values — that is, when most activity occurs.

### 5.3.2 The client application used

It makes sense that the Twitter client used to post the messages is a relevant aspect in identifying automated processes. There are many clients and methods of accessing the microblogging system (e.g. web interface, several applications, etc.). From the point of view of automation, some of these methods are easier or more convenient than others. Also, most microblogging systems have an open API, meaning that it is possible to interact with them directly. In Twitter, unregistered clients are identified as "API", while those that were registered are identified by their name.

We track the number of different clients used to post the messages, and the proportion of messages posted with each client. Cyborgs are expected to have the largest variety of clients used (as they usually post automatically from several sources). Some humans can use more than one client, for example, a mobile client and the website. Bots, on the other hand, can adhere to a single, exclusive client that is tied with their on-line presence; or may use a general client that imports messages from an RSS feed, for example.

### 5.3.3 The presence of URLs

We can make a distinction between two types of bots: information bots, which only intend to make their readers aware of something (e.g. weather forecast, TV scheduling and traffic information), and link bots, whose main purpose is to generate traffic towards their website (e.g. news, advertisements and spam). Information bots usually don't have URLs in their messages, while some link bots are capable of truncating the text of the message to make room for the URL. Most URLs shared by a bot usually have the same domain, i.e. they were all created by the same URL shortening system, or point to the same website.

Humans are also capable of introducing many URLs, but our observation reveals that cyborgs are more likely to do so; and to vary the domains of said URLs. We can observe both types of linking behaviour represented in the bot and human examples presented in the introduction, in Section 5.1.1.

We defined a feature that represents the ratio of URLs shared per total of messages written, and also keep track of the proportion of the domains associated to the URLs.

### 5.3.4 User interaction

Bots usually have one main objective that is to spread information regularly. While they may be programmed to do more complex actions (such as follow other users), automating user interaction can have undesired repercussions for the reputation of the account holder. Thus, *reblogs*[1] (to post a copy of another user's message) and *replies* (directing a message at a user) are shuned by most bots. The main exception are some spamming bots, that send several messages directed at users [GTPZ10]. To include the name of other users in the message (*mentions*) also seems to be uncommon in automated accounts.

However, a number of users also avoid some types of interaction, such as the ones previously mentioned. Therefore, while we expect this information to help identify humans, they may be less helpful in identifying bots.

With our features we keep track of the proportion of reblogs, replies and mentions, as well as the average number of users mentioned per message written.

### 5.3.5 Writing style

Our observations showed that bots usually have a fixed message template. That is, almost all bots observed end in an URL. Some have a topic at the start ("#employment", "[politics]" or "sports:"). They also appear to be written in a "less noisy" way, that is, with standard punctuation, correct capitalization, little or no abbreviations, and other stylistic choices that benefit readability. Stylistic information has been successfully used to distinguish the writing style of different people on Twitter [SSLS$^{+}$11]. Thus, we believe it to be helpful in distinguishing between automated and non automated messages since, as observed in the examples in Section 5.1.1, these users adopt different postures. The austere writing style may help with the readability, and also credibility, associated to the account, while many humans do not seem too concerned about that.

We identify the frequency (per message) of a large number of tokens, as listed below.

#### Emotion tokens

Bot operators wish to maintain a serious and credible image, and for this reason avoid writing in a style too informal (or even informal). We collect information on the use of various popular variations of smileys and "LOLs".

We also try to identify interjections. While this part of speech is culturally dependent, we try to identify word tokens that have few different letters compared with the word length — if the word is longer than 4 characters, and the number of different letters is less than half the word length, we consider it an interjection.

[1] Called "retweets" on Twitter, often shortened to "RT".

| Example | Text |
|---|---|
| 1 | Tours: Brian Wilson should retire next year `http://dstk.me/Oi6` |
| 2 | Gilberto Jordan at Sustain Worldwide Conference 2011: Gilberto Jordan, CEO of Grupo André Jordan, is the only spea...`http://bit.ly/mOOxt1` |

Table 5.1: Examples of bot Twitter messages making use of punctuation for structural purposes.

These three stylistic features were the most relevant features mentioned by Sousa-Silva et al. [SSLS+11]. Below, we can see example messages containing many emotion tokens:

- we talked before......... on twitter. **HAHAHAHAHA** RT @Farrahri: @Marcology **LOL** she smiled at me! **Hehehe**, jealuzzz not?
- **riiiiiight**.... im **offffffffffffffff**!!!!! bye bye
- RT **yessssssss**! That is my **soooong**!!!! @nomsed: You got the **looove** that I **waaaant** RT @LissaSoul: U got that BUTTA **LOOOOooooVVVEEE**!

Emotion can also be expressed through punctuation, but we include those features in the punctuation feature group, below.

### Punctuation tokens

Humans vary widely in regards to their use of punctuation. Many are not consistent across their publications. This is in opposition to bots, that can be very consistent in this regard.

Punctuation can often be used as a separator between the "topic" and "content" of the message, as can be seen on the first example on Table 5.1 . Different sources of information may structure their messages differently. Therefore one bot may use more than one separator.

We also notice that some bots publish only the headline of the article they are linking to. These articles are usually blog posts or news at a news website. Since headlines usually do not include a full stop, this feature receives a very low frequency (as seen in example 1).

Question or exclamation marks are usually infrequent in news bots, or bots looking for credibility [CMP11]. Some bots truncate the message to make room for the URL, signaling the location of the cut with ellipsis (some using only two dots). We can see an example on the second example on Table 5.1.

We measure the frequency of occurrences of:

- Exclamation marks (single and multiple);
- Question marks (single and multiple);
- Mixed exclamation and question marks (e.g. “!?!?!?!?!!?!?!!”);
- Ellipsis (normal [i.e. “. . . ”], or not normal [i.e. “..” or “....”]);
- Other punctuation signs (e.g. full stop, comma, colon, . . . );
- Quotation marks;
- Parenthesis and brackets (opening and closing);
- Symbols (tokens without letters and digits); and
- Punctuation at the end of the message (both including and excluding URLs).

**Word tokens**

This group of tokens is kept small for the sake of language independence. We begin by tracking the average length of the words used by the author. We also define features that track the frequency of words made only of consonants (that we assume to be abbreviations most of the time), and complex words. We consider complex words as those having more than 5 letters and with few repeated characters (more than half). Thus “current” (7 characters in length, 6 different characters) is a complex word, while “lololol” (7 characters in length, 2 different characters) or “Mississippi (11 characters in length, 4 different characters) do not fit the definition.

**Word casing**

Bots are usually careful in the casing they use. Careful writing aids with the image that is passed through. We measure the frequencies with which the following is used:

- Upper and lower cased words;
- Short ($\leq$ 3 letters), medium (4–5 letters) and long ($\geq$ 6 letters) upper cased words;
- Capitalized words; and
- Messages that start with an upper cased letter.

### Quantification tokens

We track the use of some numeric tokens. Dates and times are comonly used to mention events. Percentages can be more common on news or advertisements.

- Date (e.g. "2010-12-31" or "22/04/98");
- Time (e.g. "04:23");
- Numbers;
- Percentages; and
- Monetary quantities (e.g. "23,50€", "$10" or "£5.00").

### Beginning and ending of messages

Some accounts post many messages (some times all or near all their messages) using one or a small number of similar formats. This behavior is specific of bots, that automate their posting procedure. Below we can see two examples.

- **#football** Kenny Dalglish says Liverpool will continue conducting their transfer business in the appropriate manner. `http://bit.ly/laZYsO`
- **#football** Borja Valero has left West Brom and joined Villarreal on a permanent basis for an undisclosed fee. `http://bit.ly/iyQcXs`
- #football Uefa president Michel Platini claims the introduction of technology would be bad for football. http://bit.ly/jOXocn
- **New post:** Google in talks to buy Hulu: report `http://zd.net/kzcXFt`
- **New post:** Federal, state wiretap requests up 34% `http://zd.net/jFzKHz`
- New post: Kodak wins again over Apple, RIM in ITC patent ruling
  http://zd.net/kNc1rn

To determine the pattern associated with the posts, we calculate the frequency with which messages begin with the same sequence of tokens (excluding URLs and user references, that frequently change between messages). We define tokens as words, numbers, punctuation signs, emoticons and other groups of symbols that have a specific meaning.

We group all the messages by their first token. For each group with two or more messages, we store their relative proportion in a feature related to the token. We also register the 10 highest proportions found, in descending order. This entire process is then repeated, looking at the first two tokens, then the first three, and so on.

Once complete, we take note of the largest number of tokens seen, and repeat the entire process, looking at the endings of the messages.

This procedure results in a number of features that are very specific. In the case of humans we collect a relatively small number of features, as their messages can be varied. Some bots will reveal a pattern that is used for *all* their messages (e.g. see the last bot examples in Section 5.1.1). In the case of cyborgs, it is very useful to detect a number of patterns such as "I liked a @YouTube video *[URL]*" or "New Blog Post . . . ".

Below we can see examples of messages where this approach is useful. The first two messages are from a bot account, while the last two were taken from a cyborg account.

## 5.4 Experimental set-up

Our aim is to compare the level of performance provided by the five sets of features described in Section 5.3. First we create a Ground Truth by classifying a number of Twitter users manually. This data is then used to both train and test our classification system.

### 5.4.1 Creation of a ground truth

In late April 2011 we started a Twitter crawl for users in Portugal. We considered only users who would specifically state that they were in Portugal, or, not mentioning a known location, that we detect to be writing in European Portuguese. This collection started with 2,000 manually verified seeds, and grew mostly by following users that are referred in the messages. In this way our collection moved towards the more active users in the country. However, there was no guarantee that we had been collecting all the messages from any of the users.

At the moment we had over 72 thousand users and more than 3 million messages. From this set, we selected 538 accounts that had posted at least 100 messages, and 7 people were asked to classify each user as either a human, a bot or a cyborg, in accordance with our guidelines, as described in Section 5.1.1.

The annotators were presented with a series of user accounts, displaying the handle, a link to the Twitter timeline, and a sample of messages.

Since the users presented to the annotators were randomly selected, not every annotator saw the users in the same order, and the sample of mes-

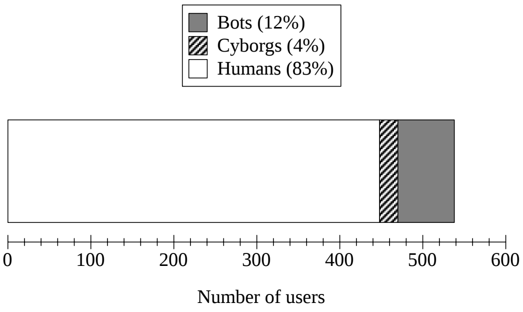


Figure 5.3: Distribution of the 538 users in the three classes.

sages for each user was also different. For each user, we considered the classification that the annotators most often attributed them. In the case of a tie, we asked the annotator to solve them before ending the voting process.

The manual classification system would only consider an account as correctly classified when the user had voted with a difference of two votes between the two most voted categories. In other words, the voting system treated a difference of one vote the same way as a tie. This was done for a number of reasons: a) it reduces the possibility of a bad vote due to an unfortunate sample of messages; b) it allows the user to be exposed to more accounts before confirming or changing his opinion about an account; and c) it allows us to better protects the results from random or misplaced votes.

To finalize the voting process, we asked one eighth annotator to solve the ties between annotators. In the end we were left with 2,721 votes, 95% of which from the 4 main annotators, that we used to calculate the agreement. Using all 538 users, Fleiss' kappa value was 0.086, that represents a slight agreement. Looking only at the 197 users that were classified by all 4 main annotators, we get a substantial 0.670 Fleiss' kappa value, showing adequate reliability in the classification.

Figure 5.3 shows the distribution of the users across all three categories. We can see that humans dominate our collection of Twitter (448), while cyborgs were the least numerous (22). In total we identified 68 bots.

### 5.4.2 The classification experiment

We randomly selected our example users from our manually classified examples. To have a balanced set, we limited ourselves to only 22 users of each type, randomly selected before the experiment. To handle the automatic classification, we opted for an SVM due to its ability to handle a large number of features. We opted for the libSVM [CL01] implementation.

For each user we selected up to 200 messages to analyze and create the features. Due to the chronological features (Section 5.3.1), we selected only

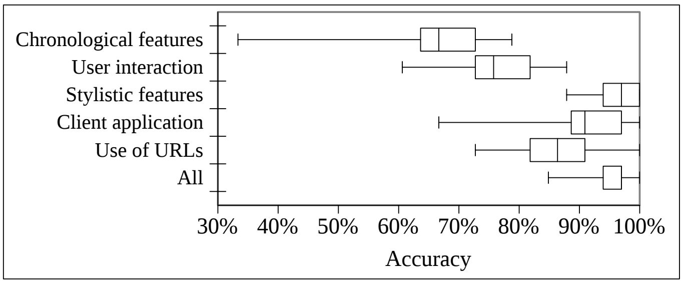


Figure 5.4: Box plot showing the results for the classification of users, using 50 2-fold cross validation runs. The limits of the boxes indicate the lower and higher quartile. The line inside the box indicates the median. The extremities of the lines represent the minimum and maximum values obtained.

sequential messages in our collection.

We used the radial basis function kernel from libSVM, allowing it to look for the parameters that best fit the data, and normalized the values of the features, allowing for more accurate results. We measured the results using the accuracy, i.e. the ratio between correct classifications and total classifications.

We opted for a 2-fold cross validation system, where we select 11 users of each type to be used in the training set, and the other 11 were part of the testing set. This allows enough testing messages to provide adequate granularity in the performance measurement, and a more reasonable number of messages to train the SVMs. We repeat each experiment 50 times (drawing different combinations in the training and testing set).

Given that we are using a balanced set of examples, we expect that a random classifier would be correct 1/3 of the times. We will be considering this as the baseline in our analysis.

## 5.5 Results and analysis

Our results are shown in Figure 5.4, detailing the minimum, lower quartile, median, upper quartile and maximum accuracy across the 50 runs.

We separate the results in two groups: the first group, that never reaches 100% accuracy, and the second group, that does.

In the first group, the user interaction features outperformed the chronological features, that had two poor runs. However, none of them shows performance similar to the other feature sets.

In the second group, the stylistic features presents the best results, with median accuracy 97%. The feature that identifies the client application also performs adequately, but twice failed 7 or 8 of the 33 examples. The URL features showed more stable results than the client information feature, but generally failed in more cases. Finally, using all the features combined yielded very good results, with 97% median (and 97% upper quartile, hence overlapping in Figure 5.4), failing once in 5 of the examples.

Over 26,000 features were generated during the experiments, most of them encode stylistic information. While in a large group they can be quite powerful (as shown), each of these features carries little information. This is in contrast with the URL, user interaction and client application features, where a small number of features can contain very meaningful information.

Most features related with the client application work almost like a database of microblog applications. That is, except for the number of different clients used, we are only recalling the identification strings present in the training messages. In the presence of an unknown client, the classifier has little information to work with. Hence the cases with low accuracy.

The features related with the URLs and with user interaction obtain information from the presence or absence of certain elements. However, in our implementation we could not encode enough information to address all the relevant cases, especially in the case of user interaction.

It is unfortunate that we are unable to compare our results with other approaches, mentioned in Section 5.2. There are three reasons for this: (i) their work has a different goal (i.e. spam detection); or (ii) the authors do not provide a quantification that we could use for comparison; or (iii) we consider that their experiment is biased (e.g. excluding some messages because they are more difficult to classify).

## 5.6 Conclusion and future work

We have shown that automatic user classification into either human, cyborg or bot — as we have defined them — is possible using standard classification techniques. With a full 97% accuracy we can say that we can be quite confident on the results of this method. However, we are unsure how motivated many of these users were to hide their posting strategy. Results may be worse in situations where the account owners take special care to hide the account automation.

Regarding the most revealing signs, as we have supposed, stylistic features can be a reliable indicator in this type of classification. In fact, they achieved results as good or better than other, more frequently used indicators of automatic activity.

Two problems arise from using the client application as the basis for user classification: first, some applications can have mixed using (as Chu et al. [CGWJ10] and Grier et al. [GTPZ10] point out); and second, dealing with the large amount of different clients is difficult. For example, we counted

2,330 different clients in our 73,848 users database (around 1 different client for every 32 users). Thus, while fast and simple, this approach does not appear to hold on its own, and should be combined with another approach.

In the future, we would like to improve our chronological features by adopting the same method Zhang and Paxson used [ZP11,GTPZ10], as our minute-of-the-hour and second-of-the-minute approach was, perhaps, too simplistic. We would also like to study the scalability of the stylistic approach, as they generate a large number of new features.

# Chapter 6

# Nationality detection

## Contents

When performing an experiment of social nature, it may be important to ensure that the population satisfies a number of criteria. A frequent criteria is the nationality, since it ensures a somewhat reasonable uniformity in a number of criteria such as culture, socio-economics and language. Nationality is also the defining boundary in a number of subjects, such as political matters, market research and advertisement.

Since the internet is often viewed as a single global community, it is difficult to determine the country of origin of a user. When consuming information from a service provider, like Twitter, there is no access to detailed information such as IP addresses, and GPS coordinates are absent from a significant number of messages. Profile information is not always available, accurate, trusted or usable for this purpose. In this work we propose using the language used by the user in their posts. Unfortunately this is a more complex problem when more than one country uses the same native language.

In this chapter we address the specific problem of detecting the two main variants of the Portuguese language — European and Brazilian — in Twitter micro-blogging data, by proposing and evaluating a set of high-precision features. We follow an automatic classification approach using a Naïve Bayes classifier, achieving 95% accuracy. We find that our system is adequate for real-time tweet classification.

## 6.1 Introduction

In this work, the specific problem we are trying to solve is identifying Portuguese users in Twitter. The official language in Portugal is Portuguese. That is also the official language in other countries, such as Brazil. However, due to cultural differences, several words evolved towards different meanings in these countries. In order to semantically understand the text, we first need to identify the nationality of the author. For example, "Nivea" is a German skin-care company very popular in Portugal, but in Brazil it is a common proper name. The text "I love Nivea" could be interpreted either as a statement of brand recognition and approval, or as a personal expression of affection for another person.

Hong et al. [HCC11] placed Portuguese as the third most used language on Twitter in 2011, with a percentage of about 9% of all messages. Portuguese is spoken mainly in Portugal and Brazil, with Brazil having approximately 20 times the population of Portugal. This shows how asymmetric the problem is: choosing a random Tweet in Portuguese, there is a 95% chance of it originating in Brazil — considering equal Twitter usage in both countries. Actually, Brazil is the second most represented country in this microblog system[1].

Since only a small fraction of users of social networks add usable information to their profiles, such as location [CCL10], identifying the nationality of the author becomes an intricate problem. Even when present, this information can be too broad (e.g. "Europe"), too specific (e.g. the street name), contain spelling errors, ambiguous names (e.g. "Lagos" — in Portugal or in Nigeria?), or present misleading information (e.g. "Hogwarts"). Thus, we are left with inferring the location of the users through indirect methods. Several solutions have been considered.

Time zones could help in locating an user on the globe. For example, by observing the regular inactive hours of users that could match their sleeping time. Unfortunately, the number of messages required to create a meaningful user profile would limit this classification to the users that

[1] http://semiocast.com/publications/2012_01_31_Brazil_becomes_2nd_country_on_Twitter_superseds_Japan

post frequently enough. Portugal and Brazil are too close geographically to allow for conclusive decisions regarding a vast percentage of users. Also, automatic posting systems, such as those from online news, have posting patterns that are less dependent on the time of day [LSO11]. Finally, when analysing some events — for example, a World Cup soccer game — the difference in time zones is less meaningful, as the users are reacting at the same moment.

Fink et al. used references to geographic features or locations [FPM+09], but since Portugal is a destination often chosen by emigrants, particularly from other Portuguese-speaking countries [Ins12], by itself this solution is insufficient in identifying the nationality of the author.

Social network information (friends and followers) is capable of providing us with an accurate location [GCCG11]. This was not used in our work because, to the extent that is possible, we wanted to rely only on information that comes with the tweets, and the network of the user does not.

We opted to tackle this problem by looking more deeply at the language used to write the messages. In many cases there is a strong affinity between languages and geographic locations, but to be truly efficient in guessing the location of a person, we need to be able to differentiate between variants of the same language [LBS+13].

Our objective is to perform this variant detection work as a classification task, where our scientific contribution lies in the set of features we propose. We distinguish between the analysis of the features (the present work) and the automatic updating of the vocabulary that assists a few of them. Since the latter task is dependent on the success of the former, we use manually constructed lists, and defer the maintenance of those referred lists to subsequent work.

We focus on features that are simple to process, work with minimal available text, and that, within the scope of micro-blogging messages, reflect differences between the European and Brazilian variants of Portuguese. These features cover common expressions, named entities, common grammatical variations, URL and writing styles. We train classification models on datasets containing up to 100 Twitter messages from each of 1400 Portuguese and 1400 Brazilian users. Although not manually annotated, our largest dataset is 56 times larger than the one used by Carter et al. [CWT13] (5000 messages distributed across 5 languages). We show that our proposed features have outperformed the traditional n-gram approach in accuracy (80%) achieving up to 95%, when using a Naïve Bayes classifier. We also show that our proposed features can be analysed faster than n-grams while still producing more accurate results.

In the next section we present related work on language identification. In section 6.3 we explain our approach for language variance identification. In Section 6.4 we describe the generation of our dataset and the experiment performed. Our results are then introduced and analysed, and our conclusions are drawned in Section 6.7. We conclude with the future steps for our work.

## 6.2 Related work

Extensive work [HBB$^+$06, GLN08] has been done in the area of automatic language identification. The most frequent approaches use character n-grams together with Naïve Bayes, or Markov Models. One of the most popular approaches is a method of profile ranking [CT94] in which the n-gram profile of a language is built, and further language classification relies on simple matching of profiles of texts of interest and the previously built reference profiles. All of the above methods attain high accuracy on moderate and large sized texts. However, the accuracy tends to drop slightly on shorter text passages.

A recent study by Gottron and Lipka [GL10] shows that Naïve Bayes, trained on 5-grams of Reuters news text in 10 different languages, and applied to the headlines of the news is able to reach 99% accuracy. The same authors achieved 81% accuracy on single-word queries. However, since news titles and news content usually have a strong affinity, high accuracy is not surprising.

Vatanen et al. [VVV10] compared profile ranking [CT94] with Naïve Bayes using different smoothing methods. The models were trained on the Universal Declaration of Human Rights in 50 languages and tested on its short cut-outs (ranging from 5 to 21 characters). They attained precision of 72.5% and recall of 72.3%, and compared it to Google's AJAX language API, with its 53.1% precision and 28.0% recall.

Distinguishing very similar languages on web pages has been identified as a difficult problem. Martins and Silva [MS05] have shown that their system, using a modified approach of Cavnar and Trenkle [CT94], is unable to distinguish between Brazilian and European Portuguese due to insignificant language differences. da Silva and Lopes [dSL06] succeeded with 98% precision in distinguishing those two variants of Portuguese using n-grams with dimensionality reduction and Quadratic Discrimination Score. They used "formal writing documents" as a corpus (e.g. official Brazilian government documents and Portuguese news), with average length of 99 lines. The work by Ljubesic et al. [LMB07] confirmed that distinguishing variants of languages is possible. They differentiated between Croatian and Serbian, using second-order Markov Models paired with a list of forbidden words. They reached precision and recall above 99%. However, Martins and Ljubesic [MS05, LMB07] used a controlled corpus of larger documents, and we are interested in short, microblog style texts.

News and web pages exhibit different linguistic properties than microblogs, that are a distinct kind of text, in part, due to their short format (e.g., 140 characters in Twitter). Tang et al. [TLC11] showed that, from a linguistic and sentiment analysis perspective, there is a difference between microblog texts and other balanced corpus (e.g. news articles). This difference validates the use of different language processing methods for microblog messages. Hong et al. [HCC11] pointed to cross-language differences in the adoption of Twitter entities such as URL, hashtags, mentions, replies and retweets. Furthermore they showed different semantic properties of languages.

Carter et al. [CWT13] used the approach of Cavnar and Trenkle [CT94] to distinguish between 5 languages on Twitter. Each language was trained over n-grams of 1000 tweets in the corpus using additional prior knowledge. Prior knowledge consisted of the languages of web pages referenced in the tweet, the language guessed of previous posts, the language used by mentioned users, the language used in the previous post in the current conversation, and the language used in other tweets containing the same tag. They achieved accuracy of 92.4% without prior knowledge and 97.4% with it.

As part of their work on latent user attributes, Rao et al. [RYSG10] studied if they could determine if Indian Twitter users, communicating in English, were from the north or south region of India based only on their messages. With 200 users in each category, they achieved their best result (77.1% accuracy) when using their sociolinguistic-feature model. These features relate to forms of expression, such as the use of smileys, ellipses, upper-cased words, “LOL”, “OMG”, and so on. Classification was done using SVM.

## 6.3 Methodology

We aim to answer the following research questions: i) how can we classify microblog users according to their variant of Portuguese (specifically European vs. Brazilian), based solely on the contents of their messages; and ii) how do our classifiers perform as we increase the number of messages per user examined?

To this end, and knowing that our ultimate goal is to identify users from (or relevant to) the Portuguese Twittosphere, we outline our approach in the following steps. First we create two sets of microblog users — one for Portugal and another for Brazil — and then proceed to sample messages from each user. We then train a classifier based on the texts written by a number of users, expecting to be able to predict the nationality of the remaining ones. The classification is supported by several groups of features that relate to six different strategies in language variant identification.

### 6.3.1 User selection

We wanted to create a large set of Portuguese and Brazilian users for our study, both to capture the variety in language and style that exists in microblogs, and to provide statistical significance to the results. Unfortunately, it would take too long to manually annotate the required number of users in a proper way. We opted to use automatic annotations, employing filters over the stated location and the social network of each user.

To improve our confidence on our annotation, we added a second source of information. Gonzalez et al. show that users tend to have geographically close followers [GCCG11]. We believe that these links are, to a degree, a user's connection to their community, and do not change, in essence, as easily as the geographic location of the user. Thus, we have greater confidence in a user that claims to reside in the same country as their friends, and should repudiate accounts that show contradictory information.

### 6.3.2 Features

We defined six feature groups: n-gram basic features and a set of five feature groups that we refer to as our *proposed* features. Some of our proposed features make use of pre-compiled lists of words, names or short expressions. These lists are meant to give an indication of the suitability of the identification strategy employed.

#### N-gram based features

In our work, n-grams set a baseline against which all other groups of features are compared.

The n-grams feature group covers all sequences of one, two and three characters in the message. N-grams are frequently used in text classification due to their simplicity and distinct results [CT94], but they can also raise the dimensionality of the feature space and extend the processing time. To avoid this problem we consider only n-grams that occur more than 0.01 × (total number of messages) times.

#### Stylistic

The style in which messages are written can help distinguish one author from another. We define a set of stylistic features based on work by Sousa-Silva [SSLS+11], since it is possible that cultural or social factors can influence the style employed by Twitter users from two different countries.

This feature group is divided into 4 segments: i) Quantitative markers, ii) Marks of Emotion, iii) Punctuation and iv) Accents. Of these, Marks of Emotion is expected to be the most relevant, since they showed the highest precision distinguishing authors [SSLS+11].

Interjections are a way of textually expressing emotion, and given the orality influence in microblogs, they can be frequent. Through observation, we noticed that Brazilians tend to express emotion using different onomatopoeias than Portuguese (e.g. the laugh "kkk" is typical of Brazil).

To identify interjections we look at words with 5 or more letters, having a low ratio of distinct characters to the word length ($\leq 0.5$). In this way we can recognise simple interjections (e.g. "hehehe") as well as more complex ones that do not follow a pattern (e.g. "hushaushuash").

### Entities

With the help of native speakers of each language variant, we put together two lists containing names that we consider likely to be mentioned in Social Media. Some of these names are idiomatic (e.g. fans of the Brazilian soccer club Fluminense are some times called "bambi"). They include regions, cities, politicians, soccer players, singers, actors, political parties, soccer clubs, big national companies, and similar entities that are frequently discussed online. The lists contain approximately 130 entries for each variant, including variations, and can be seen in Annex A.

### Word tokens

We want to determine the frequency of words exclusive to one of the two language variants. To this end we used GNU Aspell[2] and its two dictionaries for both Portuguese variants. We also took note of the words we found in neither dictionary and used them as features, since they could be vernacular mentions or entity names. In particular, we assume that words composed only of consonants are abbreviations. We rely on this heuristic instead of employing a list of abbreviations that is difficult to keep updated.

We created two lists containing expressions more popular in one of the language variants. We identified over 200 regarding Brazil, and near 80 used in Portugal, accounting for variations such as the lack of accents, common abbreviations and misspellings. For example, this list contains words such as "billion" ("bilião" in Portugal, "bilhão" in Brazil), "bus" ("autocarro" in Portugal, "ónibus" in Brazil), and expressions such as "pisar na bola" (literally "to step on the ball", but in Brazil it means to act against one's expectations).

Finally, we also count the number of times monetary references occur for both Portugal (Euro) and Brazil (Real).

[2] `http://aspell.net/`

### Grammar

Portuguese and Brazilians in some situations can show different preferences in grammar. This is noticeable in the verbal forms and the grammatical persons used. For example, in Brazil the gerund is used more often, and the second person of the singular "você" can be used in both formal and informal situations. In Portugal the infinitive is more frequent, and the second person of the singular "tu" is used in informal conversation.

Also, in Brazil the object pronoun frequently precedes the verb (e.g. "me escreve" — "[to me] write"), while in Portugal the opposite is more common (e.g. "escreve-me" — "write [to me]"). We use JSpell[3], to detect these patterns, and use a feature as a flag when one is detected.

Another distinction is the absence of the definite article before a possessive. In Brazil it is often omitted, as it is optional in the language; as in "meu texto" — "my text". In Portugal the article is frequently used: "**o** meu texto".

### URL

The last group of features is extracted from URL mentioned in the tweets. We maintain two types of features related to URL, after expansion from shortening services. The first type relates to the Top Level Domain (TLD). The national TLD for Portugal and Brazil are respectively "pt" and "br". Here, we simply count the number of URL that we find, in each message, having each of these TLD.

However, there are many websites outside of these TLD which are significantly more popular in one of the countries. For example, the hostname for the Brazilian TV station Rede Globo is "redeglobo.globo.com". To address this, we use the second type of features that counts the number of times that we find each hostname in all URL in the message.

## 6.4 Experimental setup

The tests were made in two steps. In the first step we generate the features that describe the messages in the dataset. In the second step we use a classifier to test the adequacy of the features using 5-fold cross validation.

Our goal is not to contrast different classification techniques for optimising classification accuracy, but to assess the usefulness of the sets of features we propose. Thus, the choice of classifier was secondary in our work. We opted for Naïve Bayes[4] due to its simplicity and because it is commonly used as a baseline. It also enables us to process large amounts of features quickly.

[3] `http://natura.di.uminho.pt/~jj/pln/pln.html`

[4] `http://search.cpan.org/dist/Algorithm-NaiveBayes/`

### 6.4.1 Dataset generation

Most of our message collection had been acquired with the purpose of representing Portuguese users. To ensure our current ground truth was unbiased, we used the TREC Tweets2011 corpus[5], which provides a random sample of Twitter (approximately 16 million messages) from 2011-01-24 to 2011-02-08. Our dataset was created by sampling Portuguese and Brazilian users and collecting their tweets.

In June 2011 we registered information about all the users present in the Tweets2011 corpus. In February 2012 we re-examined the users in this corpus, and discarded those that had changed the location in their profile.

We filtered out all Portuguese and Brazilian users by matching their free-form location field in Twitter against "Portugal", "Brasil" (the native spelling for "Brazil") or one of the top 10 cities (municipalities) for each country[6], excluding "Porto" (in Portugal), that is frequently mistaken for "Porto Alegre" (in Brazil). From the remaining users, we further excluded those from whom we could not retrieve more than 100 messages (excluding retweets).

In order to enable the social network filtering, we also retrieved information from the accounts that were following each of the relevant users. We selected all users with more than 10 followers, where the ratio of followers from their country exceeds those from the other country by a factor greater than 3. These conditions resulted from experimentation, and allow for an acceptable balance in the number of users in each set (2768 users in Brazil and 1455 users in Portugal), and we consider them strict enough to satisfy our labelling using an unsupervised approach. From each of these sets we randomly selected 1400 users to use in our experiments.

Finally, we used a specialised tokenizer [LSTO10] to process the messages from each user, and expanded the short URL.

After a native speaker of each language variant observed a 5% sample of each dataset, no irregularity was found in the sample of dataset "Brazil". The sample of dataset "Portugal" was more difficult to evaluate, having 6 cases that precluded a definitive conclusion. One account showed mixed spellings in words, without one of the language variants being dominant. The remaining accounts had insufficient content written in Portuguese to create an informed opinion. In conclusion, these messages are ambiguous.

### 6.4.2 Determining the number of messages

In the first experiment we wish to determine how the number of messages available influences our classification. We created several datasets with 1,

[5] https://sites.google.com/site/microblogtrack/2011-guidelines

[6] http://en.wikipedia.org/wiki/List_of_cities_in_Portugal,
http://en.wikipedia.org/wiki/List_of_largest_cities_in_Brazil

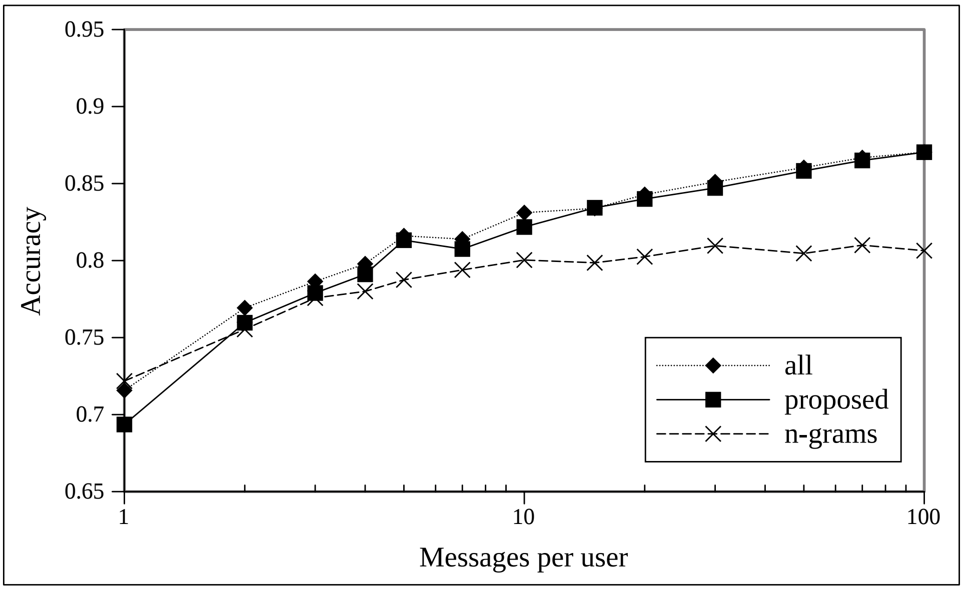


Figure 6.1: Accuracy of three feature groups as a function of the number of MPU.

2, 3, 4, 5, 7, 10, 15, 20, 30, 50, 70 and 100 messages per user (MPU).

Messages were selected randomly, and concatenated into a single text, simplifying the later processing of the text with minimal impact.

### 6.4.3 Determining the most significant features

To determine the impact of each of our proposed feature groups, we reran the experiment using 100 MPU, each time excluding one feature group. We then measured the impact this absence caused in both accuracy and execution time. This is known as a mutilation test.

## 6.5 Results

As Figure 6.1 shows, our proposed features offer nearly continuously increasing performance as more data is available. By contrast, n-gram features start to plateau fairly early (10 MPU), and fail to make use of the extra information. We can also see that our proposed features were more accurate than n-grams when exceeding 4 MPU. When using a small sample from each user (1 or 2 messages), n-grams achieve higher accuracy. Maximum n-gram accuracy was 0.80 (30 MPU), while our proposed features peaked at 0.87 (100 MPU).

In Figure 6.2 we can observe the total running times for the experiment, as a function of the sample size. This includes both training and testing

Table 6.1: Variations in accuracy and processing time of our proposed features, excluding a feature group in turn, using 100 messages per user.

| Feature Group removed | Δ accuracy | Δ time |
|---|---|---|
| Entities | -0.16% | -26.83% |
| Stylistic | +8.81% | -21.49% |
| Grammar | -0.08% | -17.18% |
| URL | -0.33% | -16.53% |
| Word tokens | -19.37% | -66.51% |

times. The time cost of both n-grams and our proposed features grow linearly with the number of MPU, but at different rates.

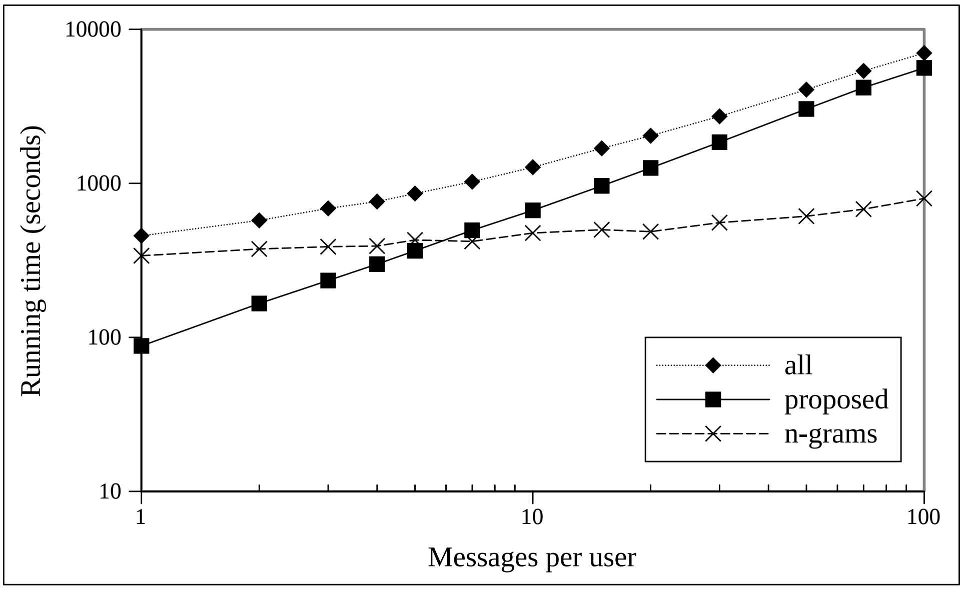


Figure 6.2: Processing time of three feature groups as a function of the number of messages used.

The results of the mutilation tests are displayed in Table 6.1. Strong emphasis should be put in the Word tokens feature group. When we removed these features, processing time decreased by 2/3, but also incurred an almost 20% accuracy loss. By contrast, our proposed features showed an improvement of almost 9% in accuracy when we excluded the Stylistic feature group, reaching the maximum value we obtained: 95%. To us, this was rather surprising since, based on previous results, we believed that these features would be adequate for this task [SSLS$^+$11] and a "community-level" style of writing could exist.

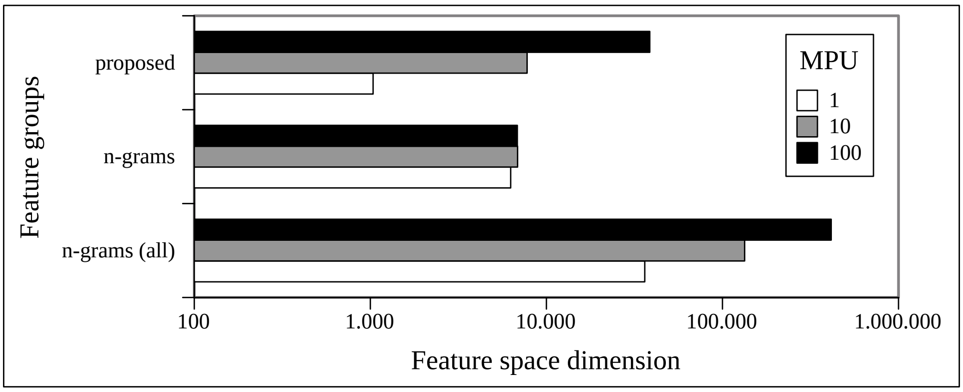


Figure 6.3: Feature space dimension of the proposed features group and limited and unlimited n-gram feature groups, using 1, 10 and 100 MPU.

## 6.6 Analysis

We should first recall that we are comparing an optimised version of n-grams with an unoptimised version of our system. In Figure 6.3 we represent the potential n-gram feature space dimension and of the feature space we actually used, regarding 1, 10 and 100 MPU. We also show the dimension of the feature space created by our five proposed groups. The number of considered n-grams is almost constant, due to the threshold described in Section 6.3.2. This allows an almost fixed execution time of the n-gram feature group. Without this threshold, the n-grams group would execute slower than our proposed features, since they accumulate many more features. This sort of feature selection could be harmful to our proposed features, since they are precision-oriented rather than recall-oriented (as n-grams). That is, they may appear infrequently, but when present they are very discriminative.

We can conclude that combining all features brought little gain in accuracy over our proposed features. The extra overhead in processing time has a minimal impact on accuracy. Compared to our proposed features, the maximum gain was inferior to 3%, when using one message per user, for over 5 times the original processing time. Compared to n-grams, 7.5% better accuracy can be obtained when using all 100 MPU, for nearly 9 times the original processing time.

The almost flat accuracy of n-grams when facing longer text samples, seen in Figure 6.1, seems to indicate that there is little gain in using a larger training set. In other words, we could say that the relative frequency of n-grams remains stable after a certain number of messages has been observed. To confirm this hypothesis, we compared the most popular n-grams when using 1 and 100 MPU. We used the Kendall tau distance [FKS03] with penalty parameter 1 to measure the ranking similarity of the 200 most frequent n-grams. The similarity measures were normalised using the maximum $\frac{1}{2}n(n-1)$, where $n$ is the number of distinct n-grams in the union of both lists being compared. In this way, 0 means that both lists are equal, and 1 indicates that the lists are in total disagreement — e.g. the n-grams are in reverse order, or no common n-gram exists.

We calculated the similarity values for the sets containing 1 message per user, and 100 MPU in each language variant. One thing that we noticed was that the Brazilian variant scored higher (greater variation) in almost every case. The most significant differences were found in 1-grams — 0.258 for Portugal, 0.302 for Brazil. This is due to URL, that can introduce seemingly random characters in the text. Each Kendall tau distance reduces to 0.061 and 0.046 respectively, when excluding URLs. Longer n-grams present greater similarity between the smaller and larger message sets. In the same order as above, similarity for 2-grams measured as 0.042 and 0.047, and for 3-grams 0.094 and 0.109. Thus, we can conclude that 1400 messages (one from each user) is sufficient to provide an accurate model of each language variant.

## 6.7 Conclusion

Our intent was to identify the variant of a language used on microblog messages, as a way to disambiguate the nationality of users. We worked on Portuguese, the official language of Portugal and Brazil (and other countries we did not use in this work), and automatically selected 1400 users referring to each country, based on information extracted from their profile and their social network. After presenting our proposed feature groups that we compared to an n-gram approach, using a Naïve Bayes classifier, we tested both approaches using several sample datasets, varying from 1 to 100 messages per user.

Our main conclusion is that lexical differences provide the best discrimination among our proposed features, and shows a promising path for classification improvement.

N-grams required minimal input to generate an adequate model, and thus, when only one or two tweets are available, this feature group provided the best accuracy. This is relevant when, for example, the user writes infrequently, or writes most messages in a foreign language, and thus we need to make a decision based on few messages.

When more messages are available, we can expect higher accuracy from our proposed features. The most accurate classification was obtained by combining Entities, Word tokens, Grammar, and URL features. In this situation we were able to reached 95% accuracy in our experiments.

As for the number of messages from each user, we notice that beyond

10 messages, each additional message contributes less to the classification. With n-grams we see negligible improvements past that point, while our proposed features continue to improve at a lower rate.

## 6.8 Future work

Some of the features presented here employ lists of vocabulary. The manual generation and maintenance of these lists is a tiresome endeavour, subject to mistakes or omissions that could greatly impact the results. The human factor limits the scalability of the system. For this reason, we intend to explore the process of generating these lists automatically. This could include external data sources, like newspapers or Wikipedia, and/or tweets.

As improvements to the classification process, we wish to test alternate classifiers, comparing their execution times and classification accuracy. We also aim to improve some features, namely in the stylistic group, and determine if they can help in user nationality classification or should be discarded.

# Chapter 7

# An analysis of obfuscation

## Contents

While most noise we have seen so far is mostly a byproduct style or a very informal form of writing, some of it is intentionally added to confuse the reader (or filtering platform). Obfuscation is thus the result of trying to raise the noise in the message so that the signal (the intended word) can pass through "unseen".

In this chapter we will describe our efforts to "see" such words and decode their true meaning.

This chapter is based on work we have previously published [LO14b, LO14a].

## 7.1 Introduction

On September 5$^{th}$, 2017 a story appeared on the Internet reproducing a humorous article that had supposedly appeared on the *Times* daily newspaper that same day and that perfectly illustrates the problem of clashing "realities" that many times profanity promotes. It was titled "War of words" and read:

> Today marks the 35th anniversary of the death of Sir Douglas Bader and I couldn't let it pass without this story of the RAF hero. He was giving a talk at an upmarket girls' school about his time as a pilot in the Second World War. "So there were two of the f***ers behind me, three f***ers to my right, another f***er on the left," he told the audience. The headmistress went pale and interjected: "Ladies, the Fokker was a German aircraft." Sir Douglas replied: "That may be madam, but these f***ers were in Messershmitts."

Confronted with the wish to tell this story, and keeping in mind the sensibility of their readers that they ought to respect, the editorial team decided to use asterisks as a form of compromise. This is a very common strategy — which we call "obfuscation" — as it avoids printing words deemed offensive by some and allows the columnist to tell their story.

Obfuscation is thus a sort of a middle road: you can no longer find the relevant word in the text, but it is still there. And of course, the primary use of obfuscation in on swearing, which seems to uniquely balance reproach and pervasiveness in language.

The work we describe here tries to improve the ability of a machine to recognise obfuscated words, and to provide the actual word excised from the text. It is curious that this task is very subjective and also difficult to be strict, meaning that even for humans this task can easily swing from trivial to a source of uncertainty and to cause disagreement in people; but that may be expected from what can be described as a creative form of communication between humans. It is even debatable if "bad obfuscations" exist, since there are no rules and "anything goes" between the extremes of "no obfuscation" and total meaningless noise, since it is supposed to be harder to understand. If User-Generated Content is occasionally compared with the "wild west" of language, then obfuscation mixes some anarchy into it.

Despite the difficulty, our results are encouraging as we were able to "decode" or "recover" many of the disguised words in our corpus back into their canonical form. We studied the obfuscation methods used by users and used that knowledge to revert the operations most frequently used on the words in a reliable manner. Our strategy is based on improvements to the Levenshtein edit distance with custom-made operations of different costs that derive from our observation of relevant data.

This chapter is divided into three parts. The first part presents background information on profanity and obfuscation. For the readers who may wonder why profanity is even a relevant thing, we dedicate the first sections of this work to address this matter. We also refer to relevant literature where possible. On Section 7.4 we begin addressing the subject of obfuscation, discussing the why and how of its uses. Section 7.5 discusses the state of the art on profanity detection (not profanity recognition or identification, as we found no previous work on this subject).

The second part of this chapter presents the data we used on our work. It starts on Section 7.6 with the detailed description of our obfuscation corpus, which allowed us to examine a Portuguese community that frequently obfuscates words. The annotation process is presented on the subsequent chapter and is followed by an analysis of the profanity found. The matter of obfuscation is addressed on Section 7.7, where the methods employed are presented in detail. We conclude this section with a revision of the most relevant information that had been presented and will be relevant to understand our work.

The third and final part of the chapter presents our work on deobfuscation and profanity recognition. It starts with Section 8.1, where we present a formal view of the obfuscation process. We then present the edit distance, explain how it works and disclose our own version of it. Section 8.3 talks about the evaluation we made of or work. This includes the methodology, the results obtained and an analysis thereof.

As usual, we close the chapter by restating the most relevant ideas we presented and draw our further line of research into the future.

## 7.2 What is profanity, and why it is relevant

Profanity, swearing, cursing or taboo words — we will be using these terms interchangeably — in the context of this work can be regarded as a form of foul, lewd, obscene or abusive language, and are usually used with offensive or vulgar intentions.

It is curious to note that the names we use today to describe this vocabulary may derive from their ecclesiastic usage to describe profane speech (profanity) or taking a sacred oath[1] (swearing) [McE06, Lju10]. Edmonds also point out that "In the past, many swear words were linked to religion." [Edm17]. Usage of certain taboo words could be invoking a deity or to give the person more power or credibility. The term "cursing" probably stems from a belief that some words hold magical power [Lju10, PT 94].

Profanity has always been a constant in both language and culture, evolving along with both and adapting to the needs of the times [McE06, Lov17]. This indicates that it dovetails with a certain need that can be considered typical in human nature, to the point where travel books include chapters on how to swear like a local [Mar14] so that people can better fit in.

When recalling the most common vocabulary used, the first thing we notice is that it is mainly of scatological or sexual nature (visceral). The Dutch, for instance, use diseases as their most grave swearing [Ges14]. Other forms of profanity are related to religion (deistic). This preference of subjects agrees with the taxonomy that Robert Hirsch proposed for modern English [Hir85], that states that "The Content of the swearing expressions are derived from the areas of taboo and stigma."

### 7.2.1 On the Internet

Before the proliferation of the Internet, mass media was the only way to reach a large number of people; and since there was a very small number of such institutions it was possible (or at least attempted) to regulate them or to ensure a somewhat uniform code of conduct. It was this uniformity that gave rise to the famous monologue of George Carlin called "Seven dirty words" [con18b], a reflection on profanity and our uneasy relation with it that starts with the seven words he said could not be pronounced on television.

The Internet changed that, and the so-called *Web 2.0* a new paradigm appeared, as users were encouraged to share their own content, their opinions and ratings, as well as to interact with each other. In this way, the platform owners became service providers, and, instead of creating new content (or in addition to doing so), took onto themselves the task of hosts or community managers. The companies cannot dissociate completely from the content published by their users, and for this reason some monitoring and code of conduct is usually present to set some boundaries. At the same time, and at the other end, other entities may wish to monitor and filter the content that is being received (such as schools and parents).

Automation is required to process a vast amount of content and to validate it as "acceptable", and it is this need that is driving most recent re-

[1] Based on passages of the Bible that condemn such actions, such as Matthew 5:34-37 and James 5:12.

search towards the machine-driven detection of offensive content. The automatic recognition of swear words is the oldest subset of this challenge, but it is still wanting.

The work presented in this chapter originates in a call for help from a research partner and media company SAPO, which was part of PT Comunicações (now assimilated into Altice). They were worried about the amount of swearing that was becoming commonplace in their sports news forums, even after a censoring filter was introduced. It turns out that swearing is common in on-line platforms, not just on SAPO; and little progress is being done in the subject of profanity detection, which is a challenge much more difficult than it appears.

## 7.3 Related work

The origin, dissemination, interpretation and use of profanity can be studied from numerous points of view, such as in psychology, linguistics, anthropology and sociology. From the computational perspective, working with profanity is commonly associated with the identification of abusive comments in User-Generated Content, with the intent of censoring them.

But profanity is also tightly related with sentiment analysis and opinion mining tasks [CDPS09], since “certain emotional states are only adequately expressed through taboo language” [JJ07]. More specifically, profanity is most often “used to express the speaker’s frustration, anger or surprise” [Jay09], or, according to the work of Wang et al. with microblogging platform Twitter [WCTS14], used mostly to express sadness and anger. It is also curious to notice that, while profanity is often associated with an idea of lower moral standards, it can also be positively associated with honesty [Edm17], as it conveys unfiltered feelings and sincerity.

How frequently is profanity written in on-line public postings? Many studies tried to quantify the prevalence of profanity in public postings on the Internet, and it appears to bear some relation with the age, gender and social class of the author [The08,WCTS14,Jay09,MP03,MX03,McE06]. Perhaps it may be possible to use this information the other way around, i.e., to try to infer some profile information about the author based on profanity use, in conjunction with other stylistic, lexical and grammatical information. While this hypothesis is strongly related with our work, we did not pursue it due to the lack of reliable data.

Some relevant studies have approached the subject of swearing, both on-line and off-line. We will be using their findings to help answer some common questions regarding this matter.

### 7.3.1 How prevalent is swearing?

The answer to this question is dependent on several factors, but we will draw upon a few works for reference.

In 1992 Timothy Jay estimated that 0.7% of words spoken in the American daily life were swear words [Jay92] (as cited by Wang et. al [WCTS14]). In 1994 Tomothy Jay is quoted as saying that up to 3% of adult conversations at work and 13% of adult leisure conversations include swearing [PT 94]. In 2003 Mehl and Pennebaker published a study where they analysed audio recordings of conversations of 52 undergraduate students across 4 days, and estimated that 0.5% of words spoken were swear words [MP03]. In that same year Grimm writes that in the US, 72% of men and 58% of women swear in public; 74% of people in the age group 18–34 year old also used such language, as did 48% of people over age 55 [Gri03]. In 2006, Subrahmanyam and Smahel observed the conversations of 583 teenagers in two chat rooms, concluding that 3% of all 12 258 utterances was obscene language, meaning one such word was used every 2 minutes [SSG06]. This same year, McEnery published an analysis of the Lancaster Corpus of Abuse (a subset of the spoken British National Corpus of spoken and written English), in which 3 to 5% of all words spoken fall within the category of "Bad Language Words" [MX04] .

As a comparison, first person pronouns (e.g. "I", "us", "our") make up about 1% of all words spoken in a conversation [MP03], and linguists do not consider them as a rare occurrence.

With this basic idea of how people use profanity in their natural surroundings, we will look at the way it is employed in three different popular on-line communities: the social networks Myspace and Facebook, the microblogging platform Twitter and the social story sharing and ranking service Yahoo! Buzz.

#### In Myspace

In 2008 Thelwall examined the pages of 8609 North-American and 767 British users of Myspace [The08]. A list of swear words was created for each of these groups, to address cultural difference. It was observed that most teenagers had swearing on their homepages, and about 15% of middle-aged people displayed strong swearing. Overall, 40% of users had profanity on their pages.

Figure 7.1 compares the likelihood of swearing appearing on the pages of people of each combination of gender and nationality. It is apparent that men swear more than their female compatriots, and that British swear more than Americans. Consequently, the British males are the most inclined to swear, but it is curious that British females curse as much as American males.

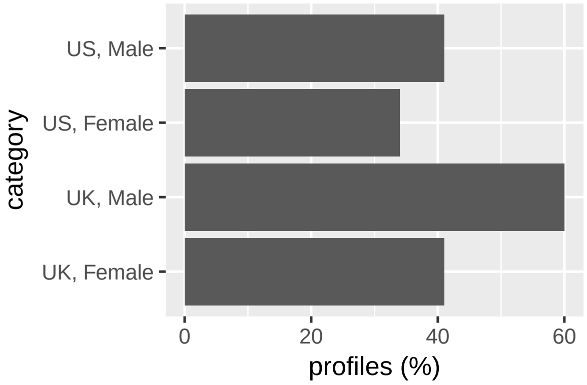


Figure 7.1: Proportion of the sample Myspace profiles containing swear words (mild, strong or very strong), grouped by gender and nationality.

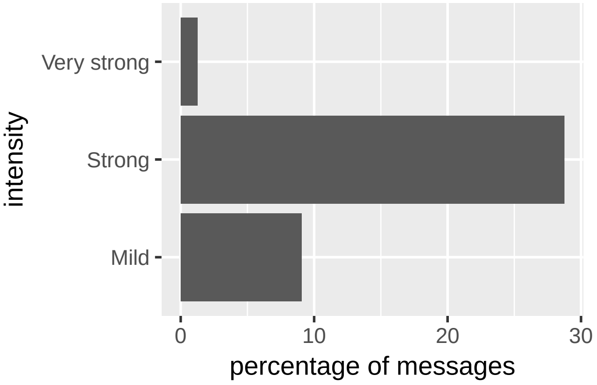


Figure 7.2: Classification of the intensity of the swear words seen on the Myspace sample profiles.

When considering the choice of swear words, the author split the relevant vocabulary into three groups: “mild”, “strong” and “very strong”. Most swearing was considered “strong”, as shown on Figure 7.2. Moderate swearing were more common on the UK (by about 2 to 3 times), with “very strong” swearing also being a much more common occurrence on British territories (2 to 5 times as much).

Thelwall’s work proceeds to focus on the “strong” and “very strong” cursing when doing word frequency analysis. The maximum percentage of such swear words in a British profile was 5%, while the maximum for an American profile was 11%. Table 7.1 shows the mean values.

In this study, the author made a good effort of seeking common word variants of swear words (for example, suffixes like “–ed” and “–ing”), portmanteau words (which were common, with a significant number of infre-

| Country | Class | Percentage |
|---|---|---|
| UK | General | 0.20 |
| | Male | 0.23 |
| | Female | 0.15 |
| US | General | 0.30 |
| | Male | 0.30 |
| | Female | 0.20 |

Table 7.1: Mean percentage of words considered strong or very strong cursing on the Myspace sample profiles.

quent ones), spelling mistakes and "deliberately unorthodox spellings," a concept on which the author neglected to provide any further detail.

**In Twitter**

More recently, in 2014, Wang et al. studied 14 million Twitter users and 51 million messages [WCTS14]. In this sample they found that 7.73% of the messages contained at least one curse word, and that 0.80% of all words was a swear word[2].

The authors pointed out that swearing was mostly associated with negative emotions. In Table 7.2 we see how often messages with swearing express the emotions sadness, anger and love, and how often messages without swearing express the same emotions. Here we can see that profanity seems to act as an expressive catalyst for the negative emotions, given the preference for these words in those situations. In addition to this, and despite the lack of value for the percentage of messages with no swearing that manifested love, curse words seem to be more strongly associated with sadness and anger than with this positive emotion. Apparently "clean" messages seem to gravitate less towards these negative feelings. This idea can be supported with the percentage of messages manifesting different emotions through swearing, which is collected in Table 7.3. Here too, the negative feelings "anger" and "sadness" appear to be the emotions more commonly expressed with cursing.

We should point out that the authors did consider alternate spelling for the words they searched for, such as "@$$". We will provide more detail on this later when we delve into more detail about this corpus, in Section 7.5.3.

[2] Due to a programming mistake, the original work reported the value of 1.15%. This value has been corrected in an updated version of the article.

| | Sadness | Anger | Love |
|---|---|---|---|
| With swearing | 21.83 | 16.79 | 6.59 |
| Without swearing | 11.31 | 4.50 | – |

Table 7.2: Percentage of tweets with and without swearing expressing certain emotions.

| Emotion | Percentage |
|---|---|
| Anger | 23.82 |
| Sadness | 13.93 |
| Love | 4.16 |
| Thankfulness | 3.26 |
| Joy | 2.50 |

Table 7.3: Percentage of tweets expressing different emotions containing swearing.

### In Yahoo! Buzz

Sood, Antin and Churchill published several works related to the study of language and on-line communities, and raise pertinent questions regarding profanity detection systems based on lists [SAC12a]. For their analysis of swearing in a collection of 6500 messages taken from Yahoo! Buzz [SCA11] — a community-based news article website — the authors chose to employ a *croudsourcing* platform called Mechanical Turk to provide their annotations for the messages. The human element provided by this solution allowed for the recognition of cursing vocabulary that could either be absent from a collection of swear words or that could have eluded the method of automatic recognition that was used. As a consequence, we trust that this operation provides a negligible number of Type II errors (false negatives), leading to a higher measurement of *recall* [SAC12a]. This is important to get a more accurate depiction of the pervasiveness of swearing online (considering that the Yahoo! Buzz community is adequately representative).

The results of this human analysis [SAC12a] showed that out of 6354 messages (146 yielded insufficient consensus) 9.4% had at least one swear word. It is unfortunate that the authors did not present details at the word level. This may be a result of the difficulties of using the croudsourcing platform for such detailed tasks, as we ourselves found *Croudflower* (the most popular croudsourcing tool) unsuitable to such minutiae tasks.

Figure 7.3 show the distribution of swearing across the different sections. Swearing was more common in political discussions and more infrequent in the sports comments. The difference between these two extremes is significant, but it also raises the question if there is any subject that is resistant to swearing.

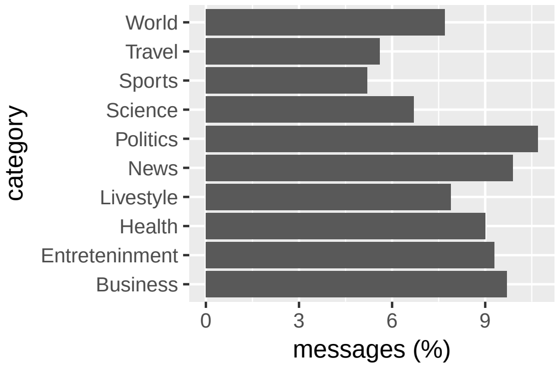


Figure 7.3: Percentage of messages in each section of Yahoo! Buzz that contained swear words.

9.4% is not too different from the 7.73% of tweets containing swear words (as seen on Section 7.3.1) — the higher recall value that we expect could justify more swear words being identified. With different years, different sources, different populations and different methodologies, we can say that the results are fairly consistent, and point towards profanity having a consistent and noticeable presence on platforms that take no actions moderating the language employed by the users.

**In Facebook**

In the 2017 study by Feldman et al. [FLKS17] 153,716 participants signed in to a Facebook application related with personality studies, granting access to their profile and messages. Of these, 73,789 fulfilled the desired criteria of using the English version of Facebook, having more than 50 status updates and more than 30 friends. 62% of the selected users were female and the average age was 25 years.

The study used the Linguistic Inquiry and Word Count (LIWC)[3] and its dictionary to calculate the profanity use rate. The mean of profanity use for the authors was 0.37%, having almost 10.8% of users not used any profanity.

### 7.3.2 Why is it difficult to censor swearing?

Part of the reason why swearing is difficult to address lies in the impossibility of compiling all the swear words in a given language. There are a number of reasons for this.

[3] https://liwc.wpengine.com/

### No definitive list

The first problem to overcome is the definition of what constitutes swearing and what are the words that need to be avoided. People tend disagree on this matter since each one has its own sensibilities. One case in particular comes from the words that used to be swear words but are, usually, no longer considered offensive anymore (for example, "damn" or "butt") [Edm17]. Not everyone agrees at the same time on the level of vulgarity that these words still hold, meaning that they may be offensive for only some people. Regardless of how comprehensive a list may be, we must also remember that new vulgar words are created all the time [TOS16].

### Word variations

In addition to collecting all the basic curse words, considering all the variations of each word can be a difficult task. For example, even for the simple word "dog" we can create a significant number of variant words (even if many do not make sense without an adequate context), that we list in Table 7.4. A collection of curse words plus all their variants is something that is impossible to accomplish in practice, especially when considering portmanteaus, which are common in English [The08] (e. g., "gunt"). This is not a situation exclusive of English, as other languages may have similar problems.

Table 7.4: Some words derived from "dog".

| | | | | |
|---|---|---|---|---|
| doggy | dogging | dogged | dogful | dogless |
| dogable | dogist | dogal | dogious | dogish |
| undogging | redogged | underdog | disdogging | dogzilla |
| dogness | dogdom | dogmageddon | dogger | doggesque |

### Different meanings

We should also keep in mind that not all swear words are self-evident profanity. Seemingly benign words can be appropriated and twisted into other connotations and through this they can be used as a negative connotation. For example, the word "gay" has changed its dominant meaning of "cheerful" to "homosexual"; "bird" acquired the slang meaning for a young woman or for an obscene gesture, "dike" can mean an embankment constructed to prevent flooding or an offensive word for lesbian, and a "fag" can be a student, a cigarette or a disparaging term for a gay man. Conversely, some say that taboo words, when used literally, should not be considered as swearing [Lju10]. Context and interpretation are very important when identifying swear words and offensive expressions — two things that are not easy to automate.

### Cultural differences

Cultural differences should also be taken into consideration. A normal word in one language can have an offensive meaning in another language. As an anecdote, Portuguese footballer Paulo Futre caused some discomfort when he enlisted in a French team, since his last name was a French slang word for sexual intercourse. French sports commentators started to consistently mispronounce his surname as "Futrè" to circumvent embarrassment.

However, most cultural differences are not clashes of different languages but the result of cultural divergences of the same language. For example, "bloody" can be a swear word in England, but not in the United States, "blow me" is an expression with no sexual connotation in England, meaning being surprised (if you blow on him, he will tip over), and "pussy" refers to cats. Similarly, the Portuguese word "veado" (stag) is used in Brazil to refer to homosexual men in an offensive way, while in Portugal it is a traditional symbol of virility. Occasionally the offensiveness of words can change even across regions within the same small country [Alm14].

### The importance of context

One other problem is that some words that are often used in a offensive way may be acceptable under the right context. One example is "bitch", which is perfectly OK when talking about dogs, but reproachable when applying it to people [SAC12a].

### Orthography

Another difficulty in recognising curse words arises from they being mainly an oral tradition. Since they are seldomly observed in written (not only in books and newspapers, but often being absent from dictionaries), the correct spelling of some less common swear words is not always obvious, and therefore they can appear in many variations — which only exacerbates the problem further. Different regions may even pronounce the words differently, making it difficult to unify them under one orthography. Some misspellings are even done on purpose, for an effect of "style" or tease (for example, the word "biatch").

The situations that we will be the trying to address with our work are in a subset of this problem. People can write the same word in multiple ways (some correctly, some incorrectly but still understandable). In Sections 2.4 and 3.1 we already mentioned the noisy nature of User Generated Content, and it spurs variations in the writing that can be taken as "noise" in the message. This makes words and messages overall difficult to process, especially automatically.

When the author purposefully uses an alternative orthography of a curse word, they may be doing so to "hide" it. This is the case of obfuscation, a cunning method used to circumvent profanity filters that we will be discussing in detail in the next section.

## 7.4 Swearing and obfuscation

Language depends on a series of symbols that convey meaning, with words being a representation for one or more ideas. Thus, when reading, we process a sequence that goes *word* → *idea*. Obfuscation adds one level of redirection, in that it references (points to or hints at) a word that is never written, existing only in the mind of the reader, but is still an essential stepping stone to understand the message. The sequence thus becomes *obfuscation* → (*word* →) *idea*.

We can say that obfuscation is simply a method used by the sender to *encode* a word in a way that the intending receiver will (hopefully) understand it, and thus avoid committing the word itself to the communication medium. But not every deviation of the word is an obfuscation, since obfuscation is a *deliberate* action, and intention is something that only the writer can declare.

If the author *misspells* a word by accident, we do not have a case of obfuscation in its strict sense. At first glance, deobfuscation (the process of resolving obfuscated writing) maintains some affinity and overlaps with the task of spellchecking (the process of correcting spelling mistakes), as they are somewhat similar text normalisation tasks. The practical difference is that deobfuscation tends to deal with a more focused vocabulary (often taboo words) and some times the author imagines an adversary or an obstacle his obfuscation must overcome — someone (children, someone from outside his group) or some thing (an automatic filter)[4]. Some clever methods to circumvent automatic profanity filters will be seen later in Section 7.7.1.

We should also make a distinction between the deliberate misspelling of a word with a *stylistic* intent and a deliberate effort to *disguise* or to *hide* it. We use the term obfuscation to refer to the latter case, where we have a conscious change of spelling of a word with the intent of making it less

[4] This competition is akin to that played between spammers (senders of unsolicited email) and anti-spam software, where obfuscation (e. g. spelling "\/1@GrA", to avoid saying "Viagra" which would raise a flag) escalated to the point where the important message is encoded into images, which are easier to process by humans than by machines Spam filters are now employing OCR technology to scan email attachments to handle such techniques [FPR06].

obvious or more difficult to understand. Again, the problem is that the *intent* of the writer is a subjective matter.

From the computational point of view, meta-subjects such as author intent is irrelevant, as all deviations it finds will be treated in the exact same way. But the question remains: *why* would someone want to make their work harder to understand on purpose? From our personal experience we have seen several possibilities:

**Pronunciation/style** The writing tries to convey the way people pronounce the words, as they think that is relevant, for instance, to hint at some typical form of speech. For example: "they don't do aaanythang out o' teh ordinary." "When I had mah operation I was incapacitated for an ageee, liek. I was also! on lots of heavy drugs aaan' it was super-fun! Anyhoohar, becaaause of all of this I ateded in bed a lot, which is to saaay friends 'n' fams broughted me lots of om noms an' I nommed 'em all up in-between zzzin'. It got borin' after a while, tho, pluses crumbs are NO (capscapscaps) joke." In the case of swearing a common use is "biatch".

Alternatively, The author may also wish to exaggerate the say some people write, as in "AMAZING!!!!!111" or "LOLZ".

**Attention grabbing** Some times the author wants to call attention to one word in the text. Often this word expresses an emotion or state of mind, as in "Ohhhhh shiiit.... one more day and then it's FRIIII-IDAAAAY."

**Differentiation** The author writes differently to display a sense of identity or general group association. "Pig latin" had been used for centuries as a method of communicating in the presence of other people, and only those "in the know" would understand it. This is just an updated and textual version of it. A common example is the so-called "leet speak" (some times written as "l33t"), where numbers replace letters and upper and lower case letters are alternating. *"Texting"*, the written communication style that emerged on cell phones is an element in the creation of a group identity and belonging to that group [Cad].

**Criticism or pun** Some times the users change words in a way to mock an entity or to make a pun. E. g. "Microsoft" written as "Micro$oft" is a common way to associate the company with money. "Dong-ald Trump" and "Bill-ary Clinton" were terms that appeared during the 2016 US election.

**Abbreviation** In order to facilitate the typing, some users may employ shorter versions of common expressions or words. These can be initialisms or created by removing letters, usually vowels. Some examples involving swearing include “fu”, “bj”, “stfu” and “ngga”.

**Acknowledging sensitivity** Since many people consider a profanity written outright as being more offensive than a *fig leaf* “asterisk” version of it (that is, “sh*t” appears more mild than the swear word properly written), the author may wish to show their respect and abstains from employing said vocabulary outright [Edm17]. Thus, the censored version should be considered more like a concession.

In these situations the words are typed differently because the offensiveness of swearing is not so much about the words but about the signals we send to each other. In the Yahoo! Buzz study [SAC12a], where no filter was in effect, 76% of the top profane words were not written correctly.

**Self-censure** When the author wishes to make a word inaccessible to some of the readers (for example, children). This relates in some way to the use of *grawlix*, the pictograms that indicate general swearing in comic books [Wal80].

Another reason to self-censure is to provide some form of deniability, a way to be able to say later “I never said that.” Consider for example a situation when a person is describing a bad situation in their former work place, or sharing some “inside knowledge” of products sold under a well-known brand, and wants to avoid saying its name to avoid trouble. They may thus refer to “O----e” instead of “Oracle”. Of course, this is all done “tongue-in-cheek”, in the same way they could say instead “a certain large and well-known database software company owned by a eccentric billionaire with a dislike for Bill Gates” and would not save the person from serious repercussions if any were intended.

Self-censure also includes circumventing automatic filtering systems, such as those aimed at preventing the use of swearing in on-line communities. Since humans are more resilient to these word alterations, these are fairly common. In an analysis of a Portuguese sports forum that started using an automatic taboo word filter [LO14b], we found that 70% of cursing was being obfuscated by the users, meaning that most changes were made to the spelling than to the vocabulary.

The general idea that we are trying to convey is that, while it is common to address obfuscation (or text that can be perceived as such) only in the context of cursing, the value of deobfuscation is not restricted to recognising “shit” written in a thousand different ways. Despite this, our work on obfuscation still focuses on the issue of swearing. The reason is simple:

- Swearing is a common event, and obfuscated swearing is also easy enough to find, thus it is easy to collect examples to work with;
- Several on-line platforms have an interest in filtering out profanity usage from the User-Generated Content they distribute, thus our work would be addressing an actual concern directly;
- The ambiguity of swear words is usually minor. Cursing if often focused on a limited lexicon and independent of the context being discussed. This is important in the annotation process.
- We have *some* possible element of comparison. We found no work dealing with non-swearing deobfuscation.

In case the reader was left wondering if encryption constitutes obfuscation, the answer is yes: encryption is an alternate form of encoding text in a way that avoids including the "proper" text, and it is an intentional effort. It is also a very complex approach to obfuscation that we will not address in this work, as it falls outside our scope, if for nothing else, because it is not a common occurrence on User-Generated Content.

### 7.4.1 Our objectives

We already mentioned how swearing may be undesirable on on-line platforms back on Section 7.3. While the main focus of many projects that deal with swear words is to simply determine if a certain message contains or not a swear word, or to say if a given word is or not considered cursing — they are profanity *detection* systems, and they can be useful to better understand and know the authors and what they are trying to express. For example, Maynard et al. address swearing on their plans for opinion mining [MBR12], and in Section 7.3.1 other examples of work related to swearing were mentioned.

With profanity *recognition* or *identification* our goal is to extend such possibilities into the realm of *obfuscated* or *disguised* swearing and to be able to say *what swear word is this (if any).* We could imagine the difference between detecting a car and identifying a car.

We are of the opinion that analysing the messages unobfuscated has advantages worth taking into account as an enabler (or facilitator) of a number of services. The following ideas could be possible:

- It would enable the preservation (recovery, we may say) of grammatical or semantic information that was provided by the user, like understanding that such word is a verb or an adjective;
- Words could be considered in their context (that is, is this swearing offensive, used as an augmentative, to convey emotion, . . . );

- While all obfuscated curse words can be labelled as generic negative words (which is incorrect sometimes), it would be more valuable if a distinction could be made between words that refer to the gender of the subject, to its sexual orientation its ethnicity and so on;
- The message could be rewritten and the profanity replaced by another word that had similar denotation but is less offensive, similar to the work of Xu and Zhu [XZ10] (much of its impact or meaning would be lost, but the entire message would not be censored);
- It could assist other systems that cared for the people's well being online, such as abuse detection (v. g. *bullying*) by providing adequate pre-processing of the messages by exposing some of the vocabulary used.

The additional knowledge that is extracted by deobfuscating messages can be particularly pertinent in the current society where so many groups are still the target of disrespectful comments and discrimination. This is a social problem that is not solvable through science or technology, but to really determine what people think (so that those concerns are addressed) we need to be able to understand the way they express themselves. Deobfuscation can improve the accuracy of automatic systems that trawl social networks to collect and interpret such impressions.

Several works have taken the challenge of identifying swearing, and we will discuss the ones that were considered relevant to the present work.

## 7.5 Works related to the use of swear words

From the computational perspective, swearing has been approached infrequently and with little progress. In their analysis of the main problems of profanity detection systems [SAC12a], Sood, Antin and Churchill stated:

> While many would argue that textual analysis is more tractable than visual content analysis, this may be in part because of a general misunderstanding about how difficult the problem of profanity detection is in real-world contexts. (...) Because of these misunderstandings, perhaps, comparatively little research has focused on detecting inappropriate text in user-generated content systems.

We will look at three types of system that typically associate with *detection* (not identification) of swear words in messages: simple list-based systems, insult or offensive message detection systems and some studies involving a significant corpus.

### 7.5.1 List-based filtering systems

The problem being addressed here can be summarised very shortly: there is some vocabulary that should be avoided; the objective is to identify any messages that may contains it so that they can be processed accordingly — usually filtered out. The onus of the entire system usually relies on the lexical component — a list of swear words for example —, as the inner working of such filters is typically quite simple. Any false negative that slips through can be remedied by adding another entry to the list.

One application of this method was to ensure more workplace correctitude, as some adjustments needed to be made when email started to be a more common tool in professional communication, with a particular emphasis given to the language used in the messages sent. US patent number 5 796 948 [Coh98] describes Cohen's invention as

> (. . . ) a method for network intercepting electronic communications or mail containing profane or offensive words, word fragments, phrases, sentences, paragraphs, or any other unit of language as may be formulated in any natural language (. . . ) and as may be formulated in any artificial language. Network profanity prevented and screened by said method includes but is not limited to vulgar language; hateful, threatening and defamatory speech; derogatory labels and terms of race, religion, gender, sexual orientation; and sexually degrading, obscene, lewd, or pornographic language.

However, regardless all the potential capabilities enunciated, this system appears to rely only on substring matching to determine what is or is not acceptable language.

During the end of the previous century, Internet reached the masses, being accessible to young people and also children. The youth needed to be protected from the unsuitable content that was present online, which lead to the creation of tools that would check if the website being accessed was suitable to its user (a form of Parental Control Software). Essentially, this was a lookup for the labels that had been attributed to the website by a human operator when they last visited it — a solution that had an obvious scalability problem. Thus, in 1999, Jacob and his team created a tool that could perform a similar task [JKR$^{+}$99]: it would look at the text of the website, search for any word in a list of inappropriate vocabulary and label the website suitable or unsuitable based on what they found. This solution, of course, had some caveats that the authors were upfront about: it only worked on text, ignoring other media like images; it applied its rules in a "blind" way, and the text may not be validated correctly (offensive text may exist without offensive words, for instance). Despite these shortcomings, this solution would scale much better than a group of humans, and could complement the existing solution.

List-based systems are used mostly because they are quite trivial to implement. The results they provide are quite poor. Sood, Antin and Churchill also wrote about their limitations [SAC12a]:

> As we have already discussed, list-based approaches perform poorly because of three primary factors: misspellings (both intentional and not), the context-specific nature of profanity, and quickly shifting systems of discourse that make it hard to maintain thorough and accurate lists.

### 7.5.2 Detection of offensive messages

There is some affinity between detecting swear words and detecting insults or offensive messages. But offensive messages and profanity are not necessarily linked. For example, the sentence "go kill yourself" is an offensive message but contains no profanity, while the sentence "what a shitty world we live in" contains swearing but is not offensive. Still, all the offensive message detection systems we will be discussing include a lexicon of profanity to help them in their task.

List based systems apply their rules blindly and directly, meaning that noise (such as that of obfuscation) can easily circumvent them. Let us look at some relevant work and how such problems were handled.

Back in 1997, Spertus described *Smokey* [Spe97], a system that detected "hostile messages", also commonly known as "flames". The author points out that this kind of messages are not the same as "obscene expressions", as only 12% of hostile messages contained vulgarities, and more than one third of vulgar messages were not considered abusive.

Smokey employs a series of fixed pre-defined rules, some of which make use of the profanity list. For example, the rules dealing with vulgarities are activated when one of the relevant swear words is found, but operate differently should a "villain" be mentioned on the same sentence. A villain is an entity usually disdained in a particular community (e.g. in a political discussion setting). Thus the aim of this tool is to deal with inter-user interaction, and not to deal with *all* offensive content.

Spertus seems to have not considered the problem of noisy text in this work, as he mentions that he removed "meaningless messages (someone randomly pressing keys)" from the collection and later, when talking about the limitation of the system, he mentions "One flame could not be recognised because the typography was unusual: 'G E T O V E R I T'."

Later, in 2008, Mahmud et al. presented their approach to hostile messages and insults using semantic information [MAK08]. Each sentence is parsed into a semantic dependency tree and the system tries to determine if they are seeing a fact being stated or an opinion. In the latter case, if an offensive word or phrase is involved, it could be an insult. The authors are upfront with the limitations of their work, stating "We didn't yet handle any erroneous input such as misplacing of comma, unmatched punctuation marks etc. at our implemented system."

In the year 2010 Razavi et al. developed a multi-level classification system for detecting abusive texts [RIUM10], employing statistical models and rule-based patterns. However the authors discarded many non-alphabetic symbols from their data, preserving only "expressive characters" such as quotation marks, punctuation signs and the hyphen.

In the same year, Xu and Zhu proposed an approach to filter offensive messages that operated at the sentence level [XZ10]. Their objective was not only to identify said messages, but also to remove the offensive content while maintaining the global integrity of the sentence. Ideally the reader would not notice the editing.

Two years later, Xiang et al. tried to detect offensive tweets by employing a bootstrapping approach [XFW+12]. In their work they considered only words composed by letters and the two symbols - and '. This, of course, leaves out much of Twitter's "rich language" like mentions and hashtags.

In the end, none of the work mentioned above addressed misspelled words, swearing or not. There is still a lot to pursue at the semantic level and the authors decided to turn their attention to the high-level matters. Deobfuscation of messages during pre-processing could help increase the coverage of such tools without changing them.

### 7.5.3 Works related to the study of swear words

On Section 7.3.1 we presented a few studies addressing online profanity. We will be revisiting them for a more detailed analysis, with particular interest on the creation of their lexicon and the methods used to recognise the swear words in the corpus.

#### Myspace

In the analysis of his Myspace homepage collection [The08], Thelwall compiled two lists of swear words to use in his studies. He used two lexicons, one for British profiles and another for American users.

The British swear word list started with the BBC's official guide, to which Thelwall added common variations (such as different suffixes) and then expanded it by generating portmanteau words. For known swear words the author seeked and added variant spellings, and finally inserted a few swear words that were still missing. Since the portmanteau words were too numerous, Thelwall discarded those present in less than 0.1% of profiles.

For the users on the USA, the author started with the "seven dirty words" (a list of words said to be forbidden to use on broadcast television by the Federal Communication Commission), and followed a process similar to the one above.

There is no information regarding the overlap of both lists, but for his main analysis the author considered the same short list of 6 very strong words, 7 strong words and 13 moderate words.

The author presented no description of a matching algorithm, and for this reason we assume that the swear words were detected by simple word comparison.

### Yahoo! Buzz

Sood, Antin and Churchill reused the data they had annotated for a project on the detection of personal insults [SCA11] (annotated through croudsourcing) on several subsequent works on swearing. They mention some problems often associated with list-based recognition systems, namely that they are easy to work around (through obfuscation), they are difficult to adapt (they cannot deal with abbreviations or mistakes), and it is difficult to tailor them to different communities [SAC12a].

Using their 6354 messages, and two lists of swear words available online (compiled for the purpose of filtering profanity), the authors showed that direct swear word matching (traditional string search) resulted in poor results. When searching for messages containing profanity, their best result (based on F1 as the relevant metric) was 0.53 Precision, 0.40 Recall and 0.43 F1 measure. This result was obtained using just one of the profanity lists and employing word stemming.

Sood, Antin and Churchill's work was annotated at the message level (that is, the annotators simply stated if the message contained a profanity), meaning that it was not trivial to isolate the swear words. The authors tried to identify the words most likely to be those responsible for the messages having been labelled as containing profanity, and noticed that only 8 of those 33 words were spelled correctly, stating:

> The remaining nineteen terms are disguised or author censored profanity (e.g., 'bullsh!t,' 'azz,' 'f*****'). Thus, a list-based profanity detection system, such as the ones evaluated in the previous section, would fail to catch twenty-five of the top thirty-three profane terms (76%) used in our data set. While these words could, of course, be added to a profanity list for future detection via a list-based system, there are countless further

Table 7.5: Performance metrics for detecting messages on the Yahoo! Buzz corpus containing profanity.

| Test | Precision | Recall | F1 |
|---|---|---|---|
| SVM | 0.84 | 0.42 | 0.55 |
| Levenshtein | 0.55 | 0.42 | 0.48 |
| SVM + Levenshtein | 0.62 | 0.64 | 0.63 |

> ways to disguise or censor words. This makes representing them in a list a significant challenge.

Regarding obfuscation, half of the swear word candidates contained non-alphabetical characters for obfuscation. As an illustration, the authors state that 40% of the uses of "@" in this collection were used in obfuscation, while the rest were part of email addresses, user mentions and other "proper" uses.

Another work by this team proposes two methods for recognising profanity [SAC12b]. The first one uses Levenshtein's edit distance [Lev66] to compare the words in the text with the words in the profanity list. This comparison is done only on words not in the profanity list and on words not in an English dictionary. The word is considered a swear word when the distance — the number of edits (additions, substitutions or deletions) — between the words is equal to the number of punctuation marks in the word, or when the number of edits is below some threshold that depends on the length of the word.

The second solution uses Support Vector Machines (SVM) with a "bag of words" approach, using bigrams with binary values representing the presence or absence of these bigrams in the message. A linear kernel is used for this task.

The most relevant performance metrics for detecting messages with cursing, based on F1 score, are presented on Table 7.5.

### Twitter

Wang et al. created a much more elaborate lexicon in their work analysing swearing in the world's largest microblogging platform [WCTS14]. They began by combining several swear word lists available online. From these words the authors kept only the ones that were in English and were used offensively. This was a manual chore, and resulted in 788 words, including graphic variations employed for obfuscation purposes.

The authors also went further in the recognition of obfuscated swearing. They devised the following algorithm: every word in the message that was not recognised by the swearing lexicon goes through a normalisation process. This process consists of removing repeating letters and then replacing numeric digits and symbols with letters that share some resemblance with them. Finally they calculate the Levenshtein edit distance between the normalised candidate word and each swear word in the lexicon, considering only the lowest. They will accept a pair as a match if the number of edits is equal to (or lower than, we assume) the number of masking symbols (i. e. noise introduced to disguise the word). Only the following seven symbols were considered as masking symbols: _ % - . # \ '. Two questions were unanswered: would "####" be considered a swear word, and why the asterisk, possibly the most common masking symbol, was not included in the list.

The evaluation was performed on a set of 1000 tweets randomly selected, and achieved a 0.99 precision score, a 0.72 recall score and a, 0.83 F1 score.

### SAPO Desporto

We consider some of our work relevant enough to include in this section, as we also created a corpus to study the use of profanity. Despite offering a more thorough description of this corpus in Section 7.6.1, for the sake of comparison with the works previously mentioned, a short summary follows.

The SAPO Desporto corpus is composed of a sample of messages taken from the leading Portuguese sports news website, that hosts an active public discussion platform for its users. 2500 randomly selected messages were manually annotated, showing that one in five messages contained one or more curse words [LO14b]. Swearing was not uniform, but it averaged one bad word every 3.2 messages, or 1% of all words written.

2500 messages does not comprise a large corpus, and is furthermore dwarfed next to some of the other bodies of work that we mentioned. So what makes it special?

The main distinction lies on the hand annotation of much finer granularity than any of the other entries. For comparison, the work of Sood, Antin and Churchill [SAC12a] on the Yahoo! Buzz corpus, which was also verified by humans, had an annotation created at the message level, i. e. "does this message contain any profanity?". Consequently, identifying the profane words was a difficult problem [SAC12b]. Wang et al. provided metrics for a sub-sample of only 1000 tweets that were hand labelled [WCTS14] and Thelwall did no evaluation of his profanity detection method on Myspace pages [The08]. As such, for practical works of evaluation, our work does not compare badly in terms of combined size and detail.

Having all profane words annotated provides us with many benefits, the most obvious of which is the possibility of more accurate measurement of the performance of our profanity recognition method, allowing the pinpointing of its shortcomings. But it can also allow us to dissociate the lexicon from the recognition process; we know that we can produce a dictionary containing all relevant profanities, and consequently any failure in recognising a word is never due to a failure in the lexicon but a problem in the recognition process. Finally, context is not unknown during the evaluation, and we can know, for example, if "bitch" is referring to a female dog and that "go *duck* yourself!" is actually cursing despite containing only upstanding dictionary words.

The final contrasting factor of this corpus is that, unlike the other corpora mentioned (or any other we could find that is focused on profanity[5]) these messages are available for use in other studies under a liberal and free license. We believe that this may contribute to further improvements in the field of profanity detection.

When each research group needs to compile and annotate their own corpus it is difficult to reproduce or compare results as variations can be introduced in the messages being used, in the lexicon or in the recognition method. By providing this annotated corpus we can work under a fixed and controlled corpus and lexicon, isolating (and thus being able to compare under the same conditions) the recognition methods.

As a closing remark, we are not oblivious to the fact that using the Portuguese language is detrimental to the "universal" adoption of this corpus. We feel that Portuguese is underrepresented in linguistic works — especially the European Portuguese — given its positioning in the most used languages in the world. Since swearing is a problem that does not exist only in English, and we were able to ensure that the results could be redistributed, we decided to contribute in this small way to our native tongue.

In the next section we will take a more technical look at this corpus, the annotation and the profanity it contains.

## 7.6 The SAPO Desporto corpus

The present work on obfuscation derives from a practical necessity from SAPO, a research partner that integrates a large media and communication company in Portugal. SAPO runs a sport news website called SAPO Desporto[6], but the swearing in the users comment sections was out of control, meaning that the filters installed were ineffective. The present work originated from the need for a better solution.

Unable to find a dataset that could be used to assist us, we created our

[5] A possible exception could be the Lancaster Corpus of Abuse [McE06], which is a subset of the spoken British National Corpus. However, we were not able to find this work available for use.

[6] `http://desporto.sapo.pt/`

own. This was a laborious process, but allowed us to set our own goals. They were as follows:

i. the messages should relate to a swearing-prone subject;

ii. the dataset needs to be of adequate size;

iii. the annotation needs to be made at the word level; and

iv. the dataset should be distributable, to be useful to others.

SAPO made available to us a sizeable sample of their user comments database and also granted the rights of redistributing an annotated corpus derived from it. To our knowledge, this represents the first dataset dedicated to the study of profanity to be freely available on-line[7], complete with detailed support files.

In the following sections we will elaborate on the details regarding the nature of the original data, how it was annotated, and what we could directly observe from it. We dedicate the largest subsection to the study of obfuscation methods, and how people use them.

### 7.6.1 Description of the dataset

Our work dataset was based on text messages published on SAPO Desporto, a sport news website, with a strong emphasis in Portuguese soccer. Soccer is known to be important for the social identity of many people in several European countries [CHFT07], including Portugal, where it can drive strong emotions in a significant portion of its population.

We were told by SAPO that sports was clearly the subject that sees the most swearing. This is in stark contrast with the Yahoo! Buzz study, that was based on data mainly from the USA [SCA11], where Sports was the category with the *least* swearing [SAC12a] (as we have shown in Figure 7.3 back on page 105).

We obtained a total of 771 241 messages posted by 218 870 accounts, related to the period 2011-04-27 to 2013-02-20. From here, we randomly selected 2500 messages, written in Portuguese, to be manually classified.

At this point we should point out, in a straightforward way, that the users of the SAPO Desporto website were not entirely free to post everything they wanted. SAPO used a simple filtering system, based on a small list of "words" that were forbidden in the comments, and rejected any messages that contained them. The users were then left with a decision: they could either not use those taboo words, or they could attempt to circumvent the filtering system in some way. From what we could see in the messages, many took this filter as a challenge, with some users even going into some

[7] `http://labs.sapo.pt/2014/05/obfuscation-dataset/`

lengths to show how cleverly they could bypass it. In the end, filtering did not end cursing — it just pushed it into disguise. As a consequence, this data became more suitable for the study of obfuscation.

The list created by SAPO contained 252 distinct entries (253 in total), including the most common profanities in the Portuguese language. We aggregated its contents into the following eight groups:

- 44 curse words,
- 85 curse word variants (plural, diminutive, etc.),
- 30 curse words written in a graphical variant (e.g. no diacritical sign, "k" replacing "c", ... ),
- 41 curse word variants written with graphical variants (e.g. "put@s"),
- 10 foreign curse words and variants,
- 12 non-curse words (e.g. "moranga"),
- 16 words with no obvious meaning, and
- 12 URL-like "words" (e.g. "mooo.com", "moranga.net", "olharestv" [sic]).

Looking at the decomposition of this data file we came to a few conclusions. First, it appears that word variants were seen as an important aspect of profanity prevention — both lexically-sanctioned variants as well as other sorts of variants, like abbreviations and soundalikes. The other conclusion is that the profanity filter expanded into a general-purpose filter, dealing with what appears to be "spam", given that almost 16% (the last three groups, totalling 40 entries) were not related to profanity or unwanted topics.

In the next section we will describe the annotation process, and then, in Section 7.6.3 the resulting lexicon will be discussed. Here we will evaluate the adequacy of SAPO's filtering choices.

### 7.6.2 The annotation

Three people worked on the annotation process. Two split among themselves the entire corpus and worked on it individually, while the third annotator coordinated the result and ensured the uniformity of the process. This arrangement precluded any inter-annotator agreement value being calculated.

In the absence of a clear definition of what constitutes profanity, each annotator used their personal judgement. We used the union of all curse words identified as the final profanity list; thus, if one annotator considered a word as being profanity, we treated it as such in the entire corpus. This task was performed by the third annotator, who also ensured that the annotation was consistent on the spelling of swear words.

We will take this opportunity to clarify the meaning of some of the terms we will be using. We use *word* in its usual term, but also as *token*, since we are working on the texts at the word level. All instances of profanity are single words that are considered taboo. In instances where offensive expressions are used, such as in "son of a bitch", we will identify the offensive word ("bitch" in this case) and ignore the rest of the expression.

Most of the time we deal with the *variety* of profanity used. When we say that we found *N* curse words it means that we observed a total of *N* *different* curse words, regardless of how many times one was repeated. This gives us a notion of the diversity of swearing. We use the term *profanity instances* to report the absolute frequency of profanity, that is, the number of times we see each swear word, to provide an idea of the popularity of that particular curse word or cursing in general.

We also use *variants* to refer to alternative spelling of words. Variants usually refer to the many different ways we see a given swear word being obfuscated (e. g. sh!t, 5hit, sh*t), but *canonical* words (that is, the correct spelling of words) also carry their own sort of variants (e. g. shitty, shitless) — this was discussed on Section 7.3.2.

The annotation is provided next to the messages in tab-separated values, where the first column contains the annotation inside brackets[8]. An example of the annotation follows.

[fudessssses=fodesses,du@ta=puta,ca@brone=cabrão,badalhoca]

In the above example, the last profanity is written correctly, while the first three are annotated in the form "*sic*=correct". The correspondence in this example is represented in Table 7.6. The original message follows the annotation unchanged.

In the 2500 messages that were manually annotated we identified 560 messages with profanities (22.4% of the messages). At the word level we counted 783 individual instances of profanity use (we disregarded repeated occurrences in the same message, when written in the same way). Of these, 544 (69.5%) were obfuscated[9].

In short, 1 in every 5 messages contained profanity (which is not claiming that none of the rest lacked offensive remarks), and if a profanity is

[8] The brackets are meant to add redundancy and help prevent editing mistakes in messages without profanity. In those situations the line would begin with white space that could be easily removed by accident.

[9] The dataset was revised since the publication of our prior work, which explains the minor deviation in values.

Table 7.6: Example classification between words seen in a message and their correct spelling.

| Word as seen in the message | Word with canonical spelling |
|---|---|
| fudessssses | fodesses |
| du@ta | puta |
| ca@brone | cabrão |
| badalhoca | badalhoca |

Table 7.7: Absolute frequency of profanities found on the 2500 messages.

| | |
|---|---|
| messages containing no profanity | 1940 |
| messages containing profanity | 560 |
| messages with 1 swear word | 404 |
| messages with 2 swear words | 108 |
| messages with 3 swear words | 31 |
| messages with 4 swear words | 15 |
| messages with 5 swear words | 2 |
| messages with 6 or more swear words | 0 |

found, there is a 30% chance that at least another one is present. Table 7.7 shows the profanity distribution, where we can see that it decreases in a near-log manner.

### 7.6.3 The lexicon

Like regular words, each swear word conveys a *meaning*, and swapping a profanity for another may change the meaning of the overall message. In order to provide a better "higher level" view of the vocabulary, we present on Table B.2 all the swear words that were identified in this corpus arranged into 40 groups, based on their base word. For clarification, this is a fine grouping where several of the groups share the same (or very similar) meaning, and not a coarse grouping based on semantics (in which case we would see significantly less groups). The numbers before the words refer to the number of different obfuscations seen for that word, which will be the subject of the next section. The Instances column shows the number of times that words in that group have been used in the corpus.

The same table shows how the Portuguese language facilitates the creation of word variations. Many of these variations are due to grammar or minor semantic nuances, some are attributable to culture (as in regional words or versions of the words), and others may be used only for obfuscation purposes (for example, the more common version of the word was being filtered). In general, these variations confound simple matching al-

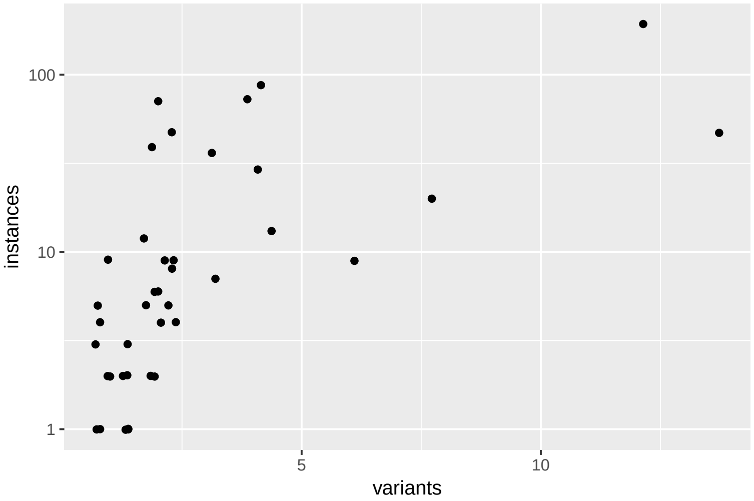


Figure 7.4: Relation between the usage of each of the 40 curse groups (instances of all variations, plotted on a log scale) and the number of variants that such group has. Some jitter applied.

gorithms that rely heavily on a dictionary to recognise words. Unsurprisingly, the most popular words also show the greater amount of variety, as Figure 7.4 makes evident.

Another observation that can be made by the readers who understand Portuguese is that most of the profanity present in the SAPO Desporto messages (as shown in Table B.2) is of scatological nature or deals with sexual matters. We also looked at the absolute frequency of each curse word in our corpus, and plotted the 10 most common as Figure 7.5, where once again we can see user preferences. These profanities are responsible for nearly 59% of all swearing observed when put together. On this list we can find references to faeces, anatomical parts, prostitution and allusion to horns — a cultural reference to a cuckold. Words related to copulation are also very common, but are spread across a large number of taboo words and variations and therefore are not included in this short list. The less common terms adhere to similar themes which, in general, align with the overall trends of swearing that were presented back on Section 7.2.

Before we delve into the obfuscation matters, we should also take a look at the filter that SAPO was employing. The profanity detection method employed by SAPO consisted of trivial case-insensitive string matching. Since we could find a few of the filtered words on the messages we annotated, written with the same spelling, we believe that we sampled some messages that predated that addition to the list. Fortunately the number of such occurrences was negligible (22 instances across 6 swear words).

It is difficult to accurately determine if the SAPO filter managed to re-

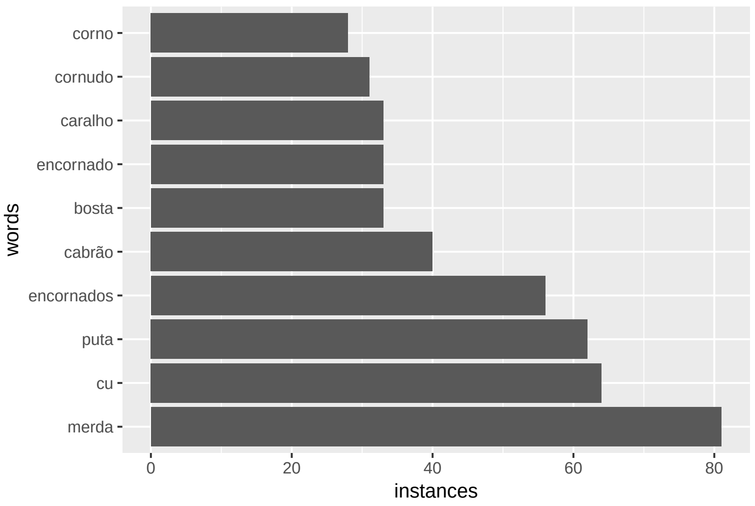


Figure 7.5: The ten most common swear words (in their canonical form) seen on the 2500 messages and how many times each was used (considering all spellings).

duce the usage of some profanity, but we are able to see what curse words remain popular *despite* being targeted by the filter.

Of the 109 profanities identified in our sample, 31 were present in the filter list provided by SAPO, which represents a coverage of about 28%. These words are represented in bold in Table B.2. Looking at the 10 most common swear words (shown in Figure 7.5) the filter includes 6 (surprisingly, "cu" [ass] was absent but several variations of it were included, which illustrates some problems with an "add as you go" approach of list creation). These 6 taboo words would account for 382 swearing instances or almost half of all swearing in the corpus — provided the filter could recognise them.

In the next section we will look closer at the methods that users employed to disguise their swearing and circumvent the filter put in place by SAPO.

## 7.7 Obfuscation

We have already seen how the canonical variations provide for a very large vocabulary. In this section we will take a look at the *obfuscation* variations, that is, the different non-standard ways in which users wrote each of the swear words they used.

While the number of uses of each profanity shows great concentration, the number of different ways in which we see words written scatters those words in many different forms. In Figures 7.6 and 7.7 we can see that many taboo words are seen only once written in a particular form. Only a handful of profane words are seen regularly with consistent orthography.

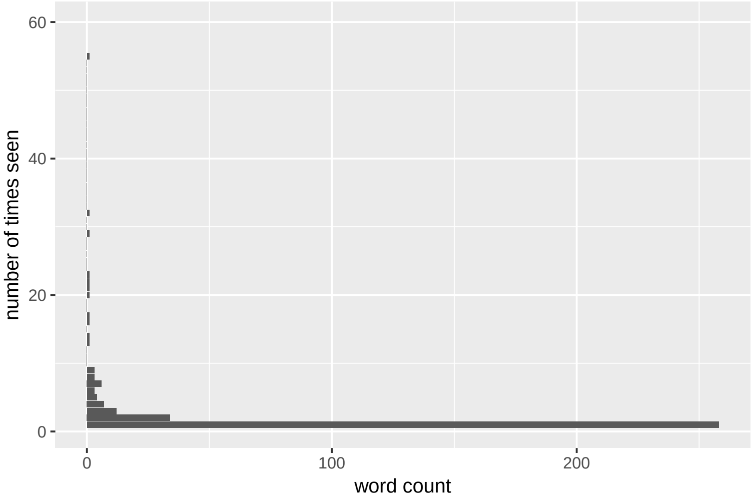


Figure 7.6: How many words are seen spelled in a certain way. Many swear words (258 to be precise) are never repeated in our sample. Some are seen twice or thrice. However, a few words are seen with the same orthography over and over. We counted eleven words that are seen more than 10 times.

In Table B.2 we precede each taboo word with the number of different spellings employed by the authors, which can include or not its correct canonical spelling (as some words are never seen written in their canonical form, even words ignored by the SAPO filter[10]). It is apparent that the words that are targeted by the filter are also the words that display the greatest spelling diversity. The few exceptions are words that one could expect to be filtered as being very similar to other censored words (such as "foda-se" and "cornos"), words that are of more regional usage ("morcão" and "morcões") and finally, as we have already mentioned, the word "cu" (ass) that is ignored by the filter while "cus" (its plural), as well as a number of other variants are in the list — which is something we are unable to justify.

It is also possible to see the filter driving the creation of alternative spellings in Figure 7.8. Here the vertical axis indicates how common different swear words are, and we can perceive a large cluster of infrequently used words at the bottom with more common words increasingly spread out at the top of the graph. This arrangement is unsurprising since Zipf's law [Zip35] is widely known to apply to language, and to profanity in particular [Pia14]. The horizontal axis of the graph shows the number of dif-

[10] As an example, "kornudos" is seen 8 times, but its correct spelling, "cornudos" is not in our sample nor is it included in the filter list provided by SAPO.

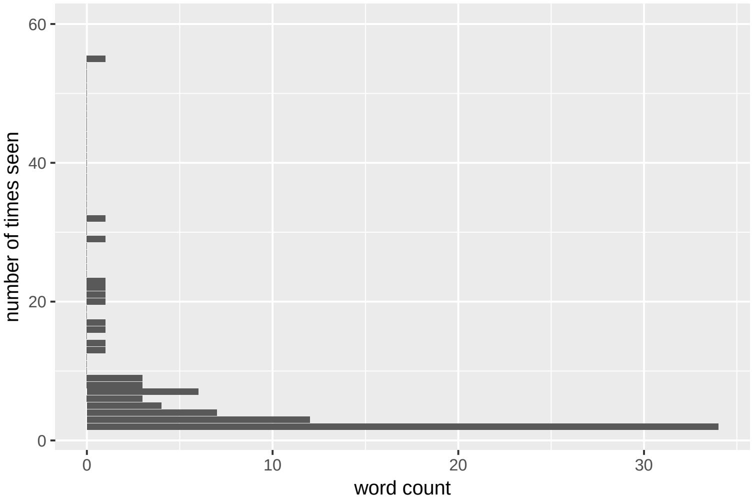


Figure 7.7: A more detailed version of Figure 7.6, containing only the words seen more than once.

ferent spellings we found for each profanity, where we notice that the more modified words are in the filter list (represented by filled circles).

In the same figure we can observe a strong correlation between the popularity of a swear word and the number of different spellings it has. Each variation tends to be seen twice in our sample, with the noticeable exception of about 5 words that deviate strongly from this trend by having a significant lower number of variations. Four of these words are absent from SAPO's filter, supporting the idea that the users felt less incentive to create alternate spellings for them[11]. The remaining word ("cornudo") is an outlier due to a strong bias towards one particular obfuscation method ("kornudo") that appears to be exceedingly popular.

But what makes an obfuscation popular? Are there any preferences in the methods used to create them? We will take a close look at the methods used by the users in our sample.

### 7.7.1 Obfuscation methods

Through manual accounting we were able to identify a total of 18 different ways in which the words we found deviated from their canonical spelling. These are described in Table B.1, with the symbols we assigned to represent them and a short description. We considered all deviations (such as misspellings) as if they were intentional.

[11] One may wonder why are there even any variations in spelling for these words if they are not being targeted by the filter. One possibility is that some users may not know if the words are being filtered and obfuscate them anyway. For our discussion about the reasons

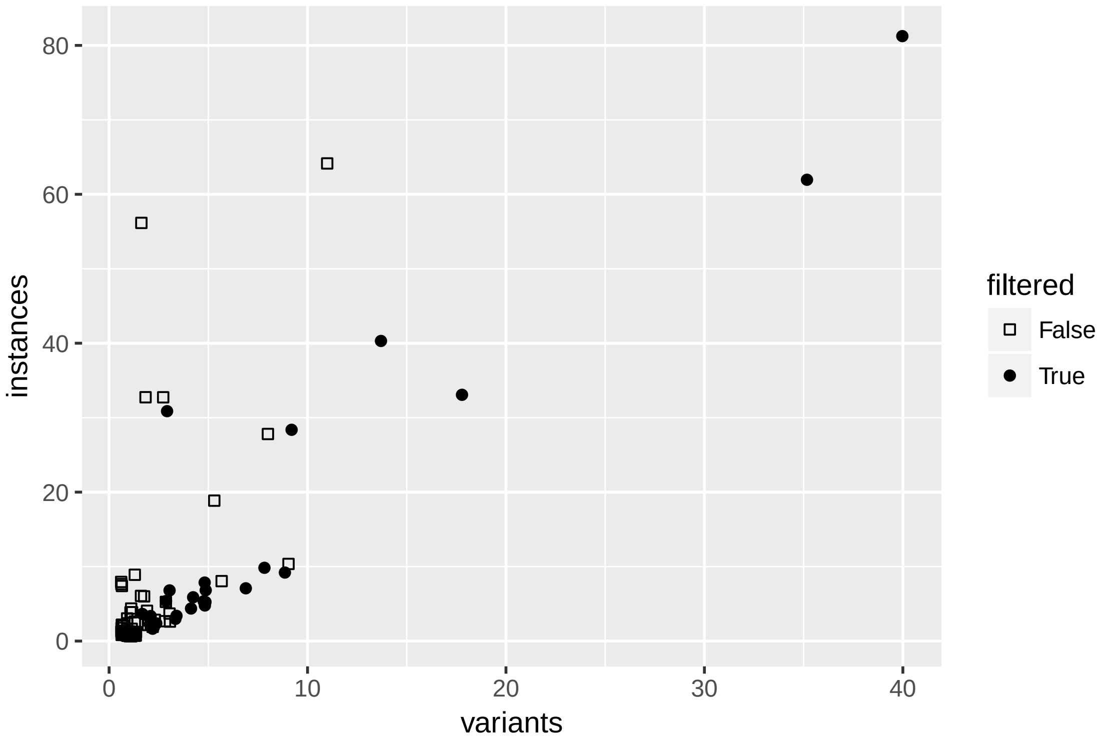


Figure 7.8: Number of uses and number of variants of each swear word, plotted with jittering. Also, the presence or absence of said swear word in the SAPO filter is also represented in the shape used in the plot.

These operations were selected to provide a descriptive view, rather than to create the smallest set of operations that could transform a word from its canonical representation. It is also possible to obtain the same result in different ways using the methods we described. For example, *Letter Substitution* can be said to be the equivalent of doing *Character removal* and *Letter addition*. But doing so would not convey the purpose of the author in the same way. Furthermore, *Letter Substitution* carries the information that we are operating on the same position in the word. We reserve the combination of the two operations to describe situations where the operations are independent.

Since terms such as "punctuation" and "symbol" may be slightly ambiguous, Table 7.8 defines the Unicode classes used to define them in a precise way. We did create an exception, though, and placed the symbol @ under "symbol" and not under "punctuation" as Unicode classification dictates. This alteration was introduced because we consider that it translates better the way this character is being used.

why text may appear to be obfuscated, please refer back to Section 7.4.

Table 7.8: General Categories from the Unicode Character Properties [Uni15] that are used to define the characters observed.

| Our set | Major class | Subclass |
|---|---|---|
| punctuation | Punctuation | connector, dash, open, close, initial quote, final quote, other |
| symbol | Symbol | math, currency, modifier, other |
| space | Separator | space, line, paragraph |
| space | Other | control |
| number | Number | decimal digit, letter, other |
| letter | Letter | uppercase, lowercase, titlecase, modifier, other |
| diacritic | Mark | nonspacing, spacing combining, enclosing |

Despite the oral tradition of swearing, it is apparent that people make an effort to use the written medium to its potential. That is to say that most obfuscation methods concern with the way words *look* and often create words with a pronunciation undefined in the process.

With the methods defined, we will proceed to see how they were used in our dataset in a detailed analysis.

### 7.7.2 Obfuscation method analysis

We should start by saying that our main goal in this study concerns obfuscation method *choice*. The number of times each method is used on each word strongly depends on word length, and provides little insight on how to reverse it; furthermore, some methods are more prone to overuse, which comes to no surprise — for example, there is no limit to the number of symbols a user can add to a word (e. g. *Repetition* and *Punctuation added*), while the number of diacritics to modify are severely limited. Thus, we do not tally reused methods nor differentiate between a modest and an exaggerated usage of methods; it all counts as just *one*.

We divided our obfuscation method analysis into two types: uses that maintain word length and uses that alter word length. We did this because obfuscations that maintain the length of a word (and thus its general shape) intact can be made much easier to decode and may employ only a subset of the obfuscation methods we described on Table B.1 (found in the Annex). Thus, the strategies used can be different.

Our analysis will focus on both types of obfuscation separately (those that maintain the length of the word and those that do not). Their distribution shown on Table 7.9 makes evident that the extra methods that the second type has available results in obfuscations that are reused less.

For each method we evaluate how many methods were combined to

Table 7.9: Distribution of obfuscations regarding modification of word length.

| Type | Distinct Words | Instances |
|---|---|---|
| Non-obfuscated | 52 | 239 |
| Obfuscation with same size | 89 | 264 |
| Obfuscation with different size | 200 | 280 |
| Total | 341 | 783 |

obfuscate the word. While we cannot claim that this is a direct indicator of how complex the obfuscation is — since some of the methods can safely be considered outside the scope of machine resolution for the time being — the number of methods used can provide an overall estimation of accessibility.

Overall, and as supported by Figures 7.8 and 7.6, with few exceptions, obfuscation reuse is not frequent, and many are even unique. While this may, in part, be due to the size of our sample, the number of obfuscation possibilities is large enough to allow this to happen on a much larger sample.

We would like to reiterate that the dataset we distribute is supplemented with the detailed information that we compile and present here.

### 7.7.3 Preserving word length

As we have just said, it is usually simpler to establish the connection between two similar words of the same size than two words of differing sizes. It also helps that the author is restricted to milder types of transformations, keeping the overall "shape" of the original word. For example, "sh*t" or "f..k".

In our annotated corpus we found 264 words obfuscated while still matching their original length, which represents nearly half of the 544 instances of obfuscation. It is curious to note that these words derive from only 89 obfuscations, which points to a noticeable reuse of obfuscations. This reuse is particularly relevant since 80% of the obfuscations we saw in our corpus were used only once. We do not know if it is a consequence of fewer obfuscation methods available of if new standards are emerging. On Table 7.10, which contains the 10 most used obfuscations, only two entries alter the length of the word.

The simpler cases we observed were obtained through the use of a single obfuscation method. Their distribution is as described in Table 7.11. At first glance it is apparent that letter substitution is, by far, the most common choice, which is easy to justify by taking a look at the list of curse words we found, in Table B.2. There we can see that many of the swear words we

Table 7.10: The ten most popular obfuscations seen in our corpus.

| Canonical profanity | Obfuscated as | Using | Number of times |
|---|---|---:|---:|
| cornudo | **k**ornudo | $^{=}L$ | 29 |
| cu | c**ù** | $^{+}Ac$ | 22 |
| cabrão | **k**abr**a**o | $^{=}L\ ^{-}Ac$ | 21 |
| corno | **k**orno | $^{=}L$ | 20 |
| puta | put**t**a | $R$ | 16 |
| cornos | **k**ornos | $^{=}L$ | 14 |
| merda | m**3**rda | $^{=}N$ | 13 |
| morcões | m**u**rcões | $^{=}L$ | 9 |
| cu | c**ú** | $^{+}Ac$ | 9 |
| bosta | **sl**bosta | $Ag$ | 8 |

found begin with the letter "c" — about 1/3 of the curse words observed —, and these include the most popular taboo words (as Table 7.10 can confirm). Replacing "c" with "k" does not alter the pronunciation of the word, and since "k" is unused in the Portuguese language[12], there almost no ambiguity possible. For this reason, replacing "c" for "k" is the most common letter substitution observed in our corpus (present in 15 obfuscations), followed by the replacement of "o" by "u", always in places where these two letters are pronounced the same (seen in 10 obfuscations in our corpus).

While authors seem to make an effort to preserve the *pronunciation* of the word when replacing one letter for another, when they use digits or other symbols to replace letters they try to maintain the *graphical representation* of the word. For example, the situations where "0" replaces "o" ($^{=}N$) and "@" replaces "a" ($^{=}S$) were responsible for half of all the instances counted in their respective categories.

Finally, changes to accents in words represent easy and simple ways to obfuscate several words. Since in very informal situations people can neglect to use the diacritical marks correctly, and there are few situations where the interpretation can become ambiguous, reading text obfuscated in this way represents little challenge for a human, as they can usually resolve any possible ambiguity. $^{=}N$ and $^{=}S$ also appear together in several obfuscations.

When combining two obfuscation methods we once again see that letter substitution is the preferred method, as displayed in Table 7.12. We already saw an example of use of the combination $^{=}L\ ^{-}Ac$ on Table 7.10, the only combination of methods that is used in more than one obfuscation. But aside from this particular word, we must say that combining methods is unusual.

[12] The letter *k* was removed from the Portuguese alphabet in 1911 and re-introduced in 1990, meaning that only new and original Portuguese words could use of it.

Table 7.11: Count of obfuscations and obfuscation instances resulting from the use of a single obfuscation method that preserved the word length.

| Method | Obfuscations | Instances |
|---:|---:|---:|
| $^{+}Ac$ | 8 | 38 |
| $^{-}Ac$ | 3 | 3 |
| $^{=}L$ | 19 | 111 |
| $^{=}N$ | 19 | 39 |
| $^{=}P$ | 2 | 3 |
| $Ph$ | 1 | 2 |
| $Pun$ | 6 | 9 |
| $^{=}S$ | 20 | 24 |
| $^{=}Sp$ | 1 | 1 |

Table 7.12: Count of obfuscation instances using two methods resulting in an obfuscation of the same length.

| | $^{+}Ac$ | $^{-}Ac$ | $^{=}Ac$ | $^{=}L$ | $^{=}N$ | $Ph$ |
|---:|---|---|---|---|---|---|
| $^{=}L$ | | 22 | 1 | | | |
| $^{=}N$ | | | | 1 | | |
| $Ph$ | | | | | 1 | |
| $Pun$ | | | | 1 | | |
| $R$ | | | | | | 1 |
| $^{=}S$ | 1 | 1 | | | 5 | |

### 7.7.4 Altering word length

In addition to the operations that we have just seen, authors may also use obfuscation methods that introduce new characters or remove them (or, in more drastic situations, go beyond simple edits). We counted 280 instances of obfuscation that resulted in a word of different length, distributed by 200 obfuscations (Table 7.9) — which points towards greater diversity than what we just saw in the previous section. Of these, 224 instances did so using only one obfuscation method (corresponding to 147 obfuscations), that we discriminate in Table 7.13. Only about 5% of these obfuscations result in shorter words.

If no other method is used, repetition seems to be the preferred choice, possibly because it makes the word stand out through exaggeration (we saw a 53 letters repetition, for example), while making the modification more noticeable — the more obvious the change, the easier it is to account while also ensuring that the change is obvious and done on purpose. Repetition leads to words only 1 character longer on the median, but due to the exaggerations that we found in the corpus, we calculated that on average 5.3 extra characters were added.

One other form of obfuscation method worthy of note, that is also used to both hide and exaggerate the word, is the addition of characters that are easy to ignore at regular intervals. Good candidates are spaces ($^{+}Sp$) and punctuation signals ($^{+}P$); for example, "w o r d", "w.o.r.d" and "w-o-r-d" are all easy to read correctly. Many obfuscations through spaces and punctuation fall into this pattern, but we also saw these obfuscation methods used in less regular fashion, as in "m erda" and "me-r.da". Such obfuscations were quite popular, but saw very little reuse unlike the repetition method.

Puns and Aggregation of words are very cultural and language dependent, as well as computationally complex. For this reasons they will not be approached more thoroughly in our work, despite being somewhat common.

When we consider the situations in which the author uses more than one obfuscation method, the result is quite different. We can see the distribution of the 52 instances (out of 49 obfuscations) in Table 7.14. Despite the lack of strong predominance of an obfuscation method, the use of symbol as a letter ($^{=}S$) and repeating letter ($R$) are seen more frequently than the other methods, even if they are not combined often.

We also accounted for the rare concurrent use of three obfuscation methods. We saw the combination $^{=}N\ Ph\ R$ three times (e.g., "f000daseeeeeee" meaning "fodase"), while $^{-}Ac\ ^{=}L\ ^{+}Sp$ were seen together once ("ka brao" written instead of "cabrão").

Table 7.13: Absolute frequency of occurrences of obfuscation using only one method that altered word length.

| Operation | Obfuscations | Instances |
|---:|---:|---:|
| $Ag$ | 15 | 29 |
| $^{-}C$ | 4 | 8 |
| $Cp$ | 4 | 4 |
| $^{+}L$ | 6 | 6 |
| $^{+}P$ | 40 | 40 |
| $Ph$ | 3 | 4 |
| $Pun$ | 15 | 26 |
| $R$ | 38 | 83 |
| $^{+}S$ | 3 | 3 |
| $^{+}Sp$ | 19 | 21 |

Table 7.14: Occurrences of obfuscation combining two obfuscation methods and altering word length.

| | $^{-}Ac$ | $Ag$ | $^{-}C$ | $Cp$ | $^{+}L$ | $^{=}L$ | $^{+}N$ | $^{=}N$ | $^{+}P$ | $^{=}P$ | $Pun$ | $R$ | $^{=}S$ |
|---:|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $Cp$ | | 1 | | | | | | | | | | | |
| $+L$ | | 2 | | | | | | | 1 | | | | |
| $Ph$ | | | | | | | | 1 | | | | 2 | |
| $R$ | 3 | | | 1 | | 3 | | 4 | | 3 | | | |
| $+S$ | | | 1 | 1 | 1 | 2 | | | | | 1 | 1 | |
| $= S$ | | 1 | 1 | | 1 | | 1 | | 6 | | 1 | 2 | |
| $+Sp$ | | | 1 | | 1 | 1 | | | 1 | 1 | | | 6 |

## 7.8 What we have learned so far

We have talked about the nature of swearing and the benefits of providing methods to recognise and process such language. We have also explained that many scientific works addressing profanity or abusive language ignored most or all cases of obfuscation because they considered them too challenging or insignificant. The numbers we obtained say that obfuscated cursing is not negligible.

Our analysis of the most relevant studies revealed two unfulfilled needs which we addressed as scientific contributions. The first of which is a corpus dedicated to the study of profanity and obfuscation. This work is available for use by other researchers, hoping that it will benefit from our cooperation. We provide detailed annotation on the obfuscated words, their canonical form, the methods that were used for obfuscation and also the relations between the curse words. Adding to the data, using the annotated corpus allows for the accounting of *false negative* classification errors and the performance of different classifiers can be compared when processing the same data.

With this work we were able to identify the most common swear words used in a sample of a popular Portuguese forum, where the native language was used. Many of those words were in the "forbidden list", disguised, showing that the introduction of filters showed little to no significant impact on the vocabulary of the users, as many circumvented the censoring through obfuscation.

From our effort we could outline a set of relevant obfuscation techniques that a profanity identification system should address, in order to find dissimulated profanity. The four more common of these are:

- replacing letters with other letters that sound the same;
- replacing letters with numbers or symbols that look like those letters;
- modifying diacritics;
- repeating letters; and
- using separation characters (usually punctuation symbols or spaces) as letter separators.

The first three of the above methods do not alter the length of the original word, while the last two increase it. Obfuscations that *reduce* word length are uncommon, and most often result from the removal of vowels. Also, not using accents when writing is a frequent occurrence — even unintentional some times — in languages that use them (in our case, a Romance language). But, even on languages that do not use diacritics, adding one can allow a user to bypass a simple profanity detection filter and still be easily understood (e.g. “shït”).

The ideal profanity identifier would address both the pronunciation of the words and their visual appearance. Since those are different problems that require different approaches, it would be very relevant if we were to understand what drives the choices of the user. Is it a matter of opportunity? That is, some characters have more lookalikes, while some sounds are covered by more graphemes? How much of it is a matter of personal preference, idiosyncratic in nature, as is the case of other types of "deviations" in informal on-line content [SSLS$^+$11]? This line of research was not pursued in this work, as it is very culture-dependable, but seems interesting to follow — that is, an analysis that adapts to the preferences of each author.

Another scientific contribution concerns with the recognition of profanities. Armed with the knowledge extracted from our corpus (and personal exposure to the subject) we try to understand what is truly being said. In the following sections we will present the results of our research on profanity recognition and identification, starting with some formalisation and the methods we employed.

# Chapter 8

# Methods of deobfuscation

## Contents

In the previous chapter we have approached the subject of obfuscation discussing the whys and the hows. But that is of little consequence to our problem (allowing better automatic processing of User-Generated Content) if we cannot reverse the noise purposefully added by the authors. This is the objective of the present chapter: to explore methods of resolving the obfuscation.

## 8.1 A formal view of the problem and our solution

Back in section 7.4 we defined obfuscation as a deviation of spelling intended to make the reading more difficult — a common consequence when modifying the way we write something. In this section we make an attempt to describe obfuscation and deobfuscation from a more formal perspective.

As we mentioned before, we are deliberately ignoring the intent condition of the obfuscation for being a speculative matter even for humans, and consequently, all deviations found are considered intentional. In addition, and without loss of generality, we will be focusing exclusively on the subject of swearing, both because it is easier to illustrate and comprehend these situations (being the dominant usage of obfuscation), and because we were drawn to the subject to address this particular usage.

We use $\Sigma$ as a symbol representing the entire *alphabet*, with $\Sigma^*$ being the set of all possible arrangements of letters (the Kleene star on $\Sigma$), which we will be referring to as *words* or *strings*. All *taboo words* are collected in $\Phi \subset \Sigma^*$, and we use $\phi$ to indicate a member of this set. We will include the *empty string* (which we designate as $\lambda$) in $\Phi$ as a special symbol to represent *"no swear word"*. In this way we can map every word in our message to a member of our swear word collection ($\Sigma^* \to \Phi$).

When obfuscating a curse word the author is replacing said word $\phi \in \Phi$ with another word $\mu \in \Sigma^* \setminus \Phi$, and hoping that the reader will resolve the correspondence. In other words, through the application of some intellectual process (that we here formalise as a function) $I$, the reader would understand that $I(\mu) = \phi$[1]. And here is our goal: to produce the best automatic approximation of $I$ that we can — which is an ambitious goal given the gap between the abilities of a computer and the human brain.

Since $I$ is a function declared as $I : \Sigma^* \to \Phi$, we intend to create our own version ($I' : \Sigma^* \to \Phi$) that tries to find the swear word that is most adequate. As we have seen in the previous section, most obfuscations are simple and preserve a significant similarity to the original word [LO14b], thus the best candidate for the deobfuscated word would be the profanity that most resembles the word we see. To find the most similar word in the set ($\Phi$) we will make use of a new function $E : \Sigma^* \times \Phi \to [0, 1]$, that returns a normalised value where 0 means that both words are equal and 1 indicates they share no resemble to one another. With this, as a starting point, we can define $I'(\mu) = \arg\min_{\phi} E(\mu, \phi)$.

It appears we did nothing more than deferring our solution, as we are now left to define function $E$. To do so we will be looking at forms of calculating the *edit distance*, which will be the subject of our next section; but for now it suffices to say that this is accomplished through a function that returns the cheapest set of operations required to transform one string into the other. Thus we have our distance function $D : \Sigma^* \times \Sigma^* \to \mathbb{N}_0$. We are only left with the task of normalising the results of $D$, which is trivial given that the maximum edit distance between two strings is the length of the longest one (assuming unitary costs). We can now define $E$.

[1] In reality it is slightly more complex than this, since the reader takes context into account, but we decided not to generalise beyond our specific needs.

$$E(\mu, \phi) = \frac{D(\mu, \phi)}{\max(|\mu|, |\phi|)}$$

There is one significant problem still persisting: just because we can find a swear word such that $E(\mu, \phi) < E(\mu, \lambda)$ does not mean that $\mu$ is a profanity. In fact, $E(\mu, \lambda) = 1$, meaning that it was difficult to consider a word "clean" (sharing one letter with a swear word was sufficient to tilt the scale towards that direction). To resolve this situation we employ a constant *threshold* value, which we represent as $\epsilon \in \mathbb{Q}^+$. This value indicates the minimum measurement of similarity that we consider relevant, and anything beyond that is ignored. It is derived from the length of the word we see written. Therefore we rewrite our function in the following way:

$$I'(\mu) = \begin{cases} \phi & \text{if } \arg\min_\phi \frac{D(\mu,\phi)}{\max(|\mu|,|\phi|)} \leq \epsilon, \phi \in \Phi \\ \lambda & \text{otherwise} \end{cases}$$

However, this threshold is still insufficient to solve the problem we mentioned. Sood said [SAC12b]:

> A pilot use of this tool found many false positives — English words with a small edit distance from profane terms (e.g. "shirt", "fitch") and first names (e.g. "mike", "ash"). To avoid these, we first verify that the target word does not appear in an English dictionary or a known list of male and female first names. If it does not appear in these three corpora, we then calculate the edit distances and judge accordingly.

In the same manner we create a set of all the *non-taboo words* (a dictionary) that we denote as $\Omega$ such that $\Omega \subset \Sigma^* \setminus \Phi$. We shall then ignore such words when we see them on the text, resulting on the new version of $I'$:

$$I'(\mu) = \begin{cases} \phi & \text{if } \mu \notin \Omega \wedge \arg\min_\phi \frac{D(\mu,\phi)}{\min(|\mu|,|\phi|)} < \epsilon, \phi \in \Phi \\ \lambda & \text{otherwise} \end{cases}$$

This solution however, comes with a caveat: should the author disguise a swear word as a non-taboo word, it will be ignored by the system (for example, writing "You think you understand, but in reality you know **shirt** about this!"). Solving such situations adequately requires very complex semantic processing, which is why this problem has been ignored in the literature. At the end of Section 8.3.6 we evaluate the impact of this condition and how common this occurrence was in our corpus.

## 8.2 The edit distance

The edit distance between two strings is a metric that indicates the minimum cost (usually equivalent to the number of elementary operations) required to transform one of the strings into the other (the edit distance is a symmetric measurement). For the sake of consistency and simple examples (and also, because we will introduce modifications that may invalidate this symmetry), we will henceforth be transforming the first argument into the second one (that is, we replace the obfuscated word we see in the text by its canonical form). The readers interested in a more significant overview of approximate string matching are referred to the survey by Navarro [Nav01].

The most common edit distance was introduced by Levenshtein [Lev66], and is commonly known as the *Levenshtein distance*. Levenshtein proposed three edit operations on individual symbols: insertion, deletion and substitution. Damerau had previously worked on symbol transposition [Dam64], which is an operation that is occasionally added as a forth elementary operation, resulting in what is known as the Damerau-Levenshtein distance [BM00]. We ignored this variant because character transposition was not seen as common in our analysis [LO14b] to justify the extra overhead, but it could be easily accommodated.

Unfortunately this algorithm proposed by Levenshtein is computationally expensive, and despite several authors independently using dynamic programming to lower its execution time [Nav01], many believe it is impossible to bring it below its current sub-quadratic time [BI14]. As we have said in the previous section, we will be comparing each word in the text to each word in the swearing dictionary, which results in a fast-growing number of distances being compared.

We will discuss the Levenshtein distance in more detail, but before we do so we will look at other alternative edit distances that may complete in less time.

The *Hamming Distance* derives from the geometric model for error detection and correction by Hamming [Ham50]. It applies over two strings of the same length and consists of the number of characters that differ. The length limitation reduces its usefulness to us and for this reason it was not used.

It was Twenty years later that Needleman and Wunsch published their distance-calculating algorithm, that was based on the concept of the *Longest Common Substring* [NW70]. This algorithm is usually employed in search operations and it is very sensitive to the order of the characters. We can describe it as the minimum number of symbols that need to be insert into the shortest chain (or part of it) to turn it into the longest one.

These two edit distances can be implemented by the Levenshtein distance: the Hamming distance is no more than the Levenshtein distance using only the substitution operation while the Longest Common Substring distance can be calculated by using only the insertion operation on the shortest string (or the removal operation on the longest one). Thus, the Levenshtein distance is the most generic one of the three.

Other string distance metrics are not based on the idea of a edit distance. For instance, the so-called *q-grams* [Ukk92] consists of the comparison of several sub-strings taken from the original chains, and bases its results on the number of matches it can find.

In our experiments we compared three edit distance algorithms: a binomial *equals* that returns only zero or one, the *Largest Common Substring* for a faster editing distance, and the *Levenshtein edit distance* for a more thorough analysis [LO14a]. Of these the Levenshtein edit distance provided the superior results.

### 8.2.1 The Levenshtein edit distance

As we just stated, the Levenshtein edit distance is supported by three operations: deletion ($d$), insertion ($i$) and substitution ($s$). These operations are defined in Equation 8.1 where $\sigma$ represents an element of our alphabet $\Sigma$.

$$\begin{aligned} d((\sigma_1,\ldots,\sigma_{n-1},\sigma_n)) &= (\sigma_1,\ldots,\sigma_{n-1}) \\ i((\sigma_1,\ldots,\sigma_n),\sigma_m) &= (\sigma_1,\ldots,\sigma_n,\sigma_m) \\ s((\sigma_1,\ldots,\sigma_{n-1},\sigma_n),\sigma_m) &= (\sigma_1,\ldots,\sigma_{n-1},\sigma_m) = \\ &= i(d((\sigma_1,\ldots,\sigma_{n-1},\sigma_n)),\sigma_m) \\ & \quad \sigma_i \in \Sigma, \forall i \in \mathbb{N} \end{aligned} \tag{8.1}$$

The most common algorithm that computes the edit distance will operate over the increasingly longer prefixes of both chains in order to find the shortest possible sequence of edits that can transform the first chain into the second one. Please note that many algorithm implementations, for the sake of efficiency, discard the sequence of edits and compute only the minimal edit cost. For our purposes the edit operations are paramount.

The Levenshtein distance is usually computed through a matrix or table. The algorithm computes all the edit distances of all pairs of prefixes recursively using Equation 8.2, where $i$ and $j$ start as the lengths for strings $a$ and $b$ respectively. $\mathbb{I}$ is the equality function defined between two symbols, as defined by Equation 8.3.

For illustrative purposes, our examples consist in transforming the word "seeds" into the word "looser". Recursively calculating the edit distance for these words using Equation 8.2, and disposing the results into a matrix where each cell holds the transformation cost between each pair of prefixes, we arrive at Figure 8.1. This matrix is the basis of the Wagner–Fischer algorithm [Nav01].

| | | l | o | o | s | e | r |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| s | 1 | 1 | 2 | 3 | 3 | 4 | 5 |
| e | 2 | 2 | 2 | 3 | 4 | 3 | 4 |
| e | 3 | 3 | 3 | 3 | 4 | 4 | 4 |
| d | 4 | 4 | 4 | 4 | 4 | 5 | 5 |
| s | 5 | 5 | 5 | 5 | 4 | 5 | 6 |

Figure 8.1: Matrix of the Levenshtein edit costs between the prefixes of the words “seeds” and “looser”.

$$lev_{a,b}(i,j) = \begin{cases} i \text{ if } j = 0 \\ j \text{ if } i = 0 \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+\mathbb{I}(a_i,b_j) \end{cases} & \text{otherwise} \end{cases} \tag{8.2}$$

$$\mathbb{I}(a,b) = \begin{cases} 0 & \text{if } a = b \\ 1 & \text{otherwise} \end{cases} \tag{8.3}$$

The lower right cell tells us that the final edit distance is 6. This is also the length of our longest word, hence the maximum edit distance possible. This means that the words are very distinct.

As we are more interested in *how* one word is transformed into the other, having the maximum edit cost means only that all transformation arrangements are all equally bad. Table 8.1 shows four different transformation possibilities, all resulting in the same final cost.

### 8.2.2 A different way to see the Levenshtein edit distance

People do not obfuscate based on standard edit costs, and refer instead to other preferences, some of which we have already seen (in Section 7.7.1). We thus begin our search for an improvement to the basic Levenshtein method by following the series of operations mentioned in Table 8.1 across the matrix that we see in Figure 8.1. This provides a way to see how the edit cost progresses with each operation.

Table 8.1: Multiple options in transforming "seeds" into "looser" with edit cost 6 (the minimum possible for these words). The letter style indicates the operation: *inserted*, ~~deleted~~, replaced, **kept** (i. e. replaced by the same character with no cost).

| text | comment | text | comment |
|---|---|---|---|
| seeds | | seeds | |
| *loo*seeds | insert "loo" | ~~s~~eeds | eliminate the starting "s" |
| *loo***se**eds | maintain "se" | ~~s~~loos | replace "eed" with "loo" |
| *loo***se**rds | replace "e" with "r" | ~~s~~loo**s** | keep the "s" |
| *loo***se**r~~ds~~ | remove the extra "ds" | ~~s~~loo**s***er* | add "er" |

| text | comment | text | comment |
|---|---|---|---|
| seeds | | seeds | |
| *l*seeds | insert an "l" | loose | replace all letters |
| *l*ooser | turn "seeds" into "ooser" | loose*r* | add the final "r" |

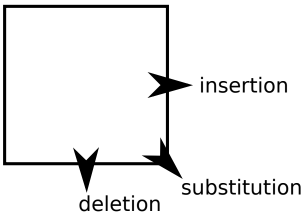


Figure 8.2: How each of the three directions translates into an edit operation.

The top left cell, that compares both empty strings, always starts with zero. From there we can move to another adjacent cell (select another cell as the relevant one) across three directions, according to the prefixes we decide to compare and the operations we employ. Figure 8.2 shows how directions and operations relate.

As described previously by Equation 8.1, when we do an insertion, we take the next letter of the target string; when we do a deletion, we ignore or discard a letter from the source string; and when we do a substitution we do both operations. Or another way of perceiving these operations is imagining that we start with the source string and we try to transform it into the target string, operating on a *cursor* whose position is defined by the target prefix being compared. The cursor separates the target string (to its left) from the source string (to its right). Figure 8.3 provides an illustration of

| | | l | o | o | s | e | r |
|---|---|---|---|---|---|---|---|
| | ·seeds | l·seeds | lo·seeds | loo·seeds | loos·seeds | loose·seeds | looser·seeds |
| s | ·seed | l·eeds | lo·eeds | loo·eeds | loos·eeds | loose·eeds | looser·eeds |
| e | ·see | l·eds | lo·eds | loo·eds | loos·eds | loose·eds | looser·eds |
| e | ·se | l·ds | lo·ds | loo·ds | loos·ds | loose·ds | looser·ds |
| d | ·s | l·s | lo·s | loo·s | loos·s | loose·s | looser·s |
| s | · | l· | lo· | loo· | loos· | loose· | looser· |

Figure 8.3: The cursor makes evident the prefixes being compared and acted upon.

this movement, where each cell shows the word being worked on and the dot represents the editing cursor. As with the Wagner–Fischer algorithm, we start from the top left cell and move down and right towards the final position where we conclude our transformation.

In Figure 8.4 we represent the paths of the transformation examples provided in Table 8.1. The zero-cost operations occur when the source and target letters are the same in a substitution, and we signalled those four possible occurrences with a small zero.

Other paths (or sequences of operations) could be taken, including some that would accumulate a larger edit cost — for example, starting with the deletion of all 5 letters of “seeds”. It may just happen that some non-optimal processes are more natural to humans or attuned to their preferences. We would like to explore this possibility, and to do so we will first restructure the matrix that we use.

### The graph map

We propose that we look at the Levenshtein matrix more like a graph. Instead of using the cells to store the costs — meaning “when comparing these two prefixes the lowest possible edit cost is this” or “upon reaching this situation the lowest possible cost accumulated is such”, which can account only for one value —, we could interpret the cells as *nodes* that represent *state*, i. e., “which prefixes have we computed so far” or “what is the active state that is being worked on?” Edit operations that lead to another active cell will be interpreted as *edges* that lead to another node. Thus, each

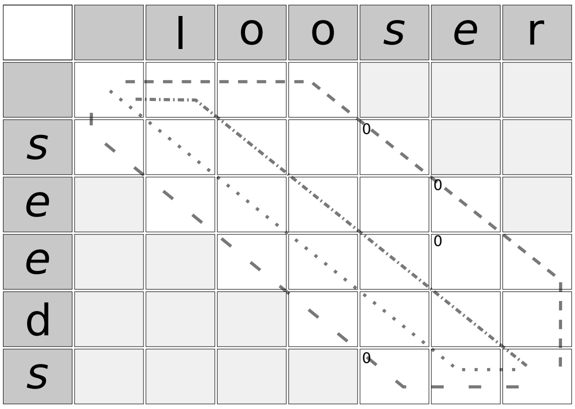


Figure 8.4: Adaptation of the Levenshtein matrix table from Figure 8.1 to show the four paths presented on Table 8.1. Zeros indicate operations that could be performed without increasing the total cost of the path. Darker cells represent positions that, if crossed, would make a 6-cost path impossible — this does not mean that every path that avoids them will have the lowest cost.

node will have up to three edges leading to it, and up to three edges leading out of it.

We still wish to maintain similarity with the original matrix, thus we decided to represent the nodes/cells as hexagons as shown in Figure 8.5. This has the advantage of all operations being displayed in the same way, meaning that the operation substitution is no longer displayed as a diagonal movement but has its own geometric edge, like the other operations had.

A desired consequence of using three edges for movement/operation is the possibility of using them to represent the operation cost. For example, Figure 8.6 shows similar information to Figure 8.1 (excluding the costs, which are path-dependent), but now zero-cost operations (substitutions of the same letters) are made evident by omitting the relevant edge.

If our objective was to find the path (the sequence of operations) with the lowest cost, we would be able to employ a number of algorithms to do so, such as the Dijkstra [Dij59] algorithm. The results would be the same.

### New operation costs

At this point we should point out that the costs associated to operations need not be limited to one or zero. Any non-negative cost can be attributed to each operation. We said that users like to add extra characters in their obfuscations (back in Section 7.7.1), so we can adjust the costs to reflect that preference. That would mean that insertion operations are cheaper than

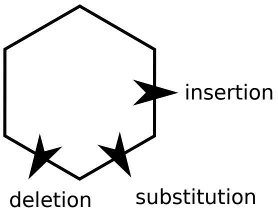


Figure 8.5: The three directions of movement over a node. The three edit operations are associated with edges.

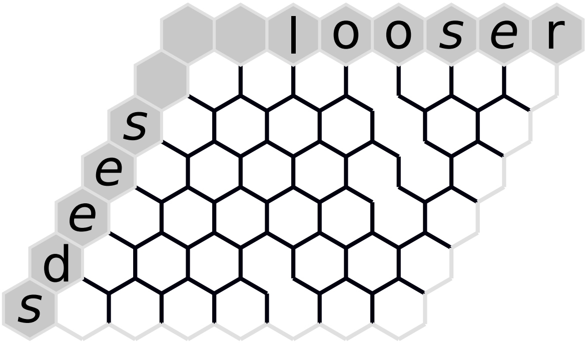


Figure 8.6: Map of the Levenshtein edit distance for transforming “seeds” to “looser”. Notice the zero-cost transitions (the four missing edges).

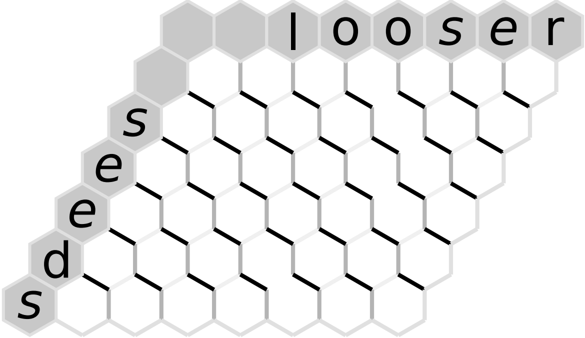


Figure 8.7: The map now shows different costs for the operations. From the most expensive (darker) to the less expensive (lighter) we have: deletion, insertion and substitution (which could cost zero in some cases).

deletions, but still more expensive than substitutions (because we also saw that authors like to maintain word length). The result of this example is illustrated in Figure 8.7.

Let us remember that obfuscation methods can be influenced by personal preferences and the environment itself (users may choose to employ some method that is popular in the discussion forum, when typing on mobile, . . . ). This means that no set of costs can be considered as ideal, since what works well for venue $A$ may do much worse in venue $B$ or in venue $A$ after a year has gone by. We need some way to *adapt* to different environments instead of accepting a set of universal costs for the edit operations. Further opportunities for adaptation exist, with finer granularity, and will be presented on the next section and as future work at the end of this chapter.

**Determining operation costs** It makes sense to base the cost of each edit operation on the number of times it needs to be employed to deobfuscate words. Thus, if the author is inserting extra letters to disguise their words, the deletion operation should be cheaper than the insertion. But there is a problem determining the frequency of each operation: there are likely multiple ways of editing one string into the other one — remember that each substitution can be replaced by an insertion and a deletion. We can see an example in Table 8.2, where approach $(I)$ gives equal weight to operations $s$ and $i$, while approach $(II)$ results in a dominating insertion.

Given a number of edit strategies we want to calculate the usefulness of each edit operation. Let us suppose that we have $n$ pairs of swear and regular words $(\phi, \omega) \in \Phi \times \Omega$ that we call our sample $A$. Each pair may

Table 8.2: Comparison of two string transformation sequences, and how they result in different operation frequencies.

| Situation: | $(I)$ | $(II)$ |
|---|---|---|
| Transformation "A" → "B": | $s$ | $d, i$ |
| Transformation "C" → "CD": | $i$ | $i$ |
| Relative frequencies: | $s = 1/2$ | $d = 1/3$ |
| | $i = 1/2$ | $i = 2/3$ |

have one or more edit paths that can transform $\phi_i$ into $\omega_i$. We can represent all of these as $P_i^{\phi_i,\omega_i}$, which we shall simplify as simply $P_i$. Each path is simply a sequence of edit operations. If we had only a single path per pair of words, calculating the relative frequency for each operation would be trivial and we would get only a single result. With a number of paths per pair of words we need to consider every permutation ($\prod_{i=1}^{n} |P_i|$), which leads to a number of possible results that grows too quickly with the sample size.

Fortunately we don't need to compute all possibilities and can reduce this value to one that is maximised by $3n$ since we care only for the extreme frequencies of each edit operation. We are interested in finding out how much work would be done by each of the operations should it be preferred over all the others. The end result is just three scenarios or less for each pair of words — in none of which we have the guarantee that the privileged operation has the dominant frequency (we can see this happening in point $(I)$ in Table 8.2, where we maximise the usage of $s$, and in point $(II)$ where we maximise the usage of $d$).

Once obtained the frequencies, they are translated into new operation costs. This is a trivial $\mathbb{Q}^3 \rightarrow \mathbb{N}^3$ function where the only requirement is that the proportions between the elements are preserved up to a reasonable degree. We are now left to determine which one is better suited.

It should now be determined which cost set has the greater discriminatory power when classifying disguised curses and innocuous words. For this we employ a second sample of annotated obfuscations $B_o$ and of non-obfuscated "clean" words $B_c$ that we submit to a *fitness function*. Such function is based on the median of all normalised distances of each set, with $\tilde{B_o} = \text{median}(E(x)),\ x \in B_o$ and conversely $\tilde{B_c} = \text{median}(E(x)),\ x \in B_c$. The best candidate is that which maximises $\tilde{B_c} - \tilde{B_o}$.

**Calculating the threshold** Finally we miss only the parameter $\epsilon$, which is the threshold, the cost that, if exceeded, we do not consider the word an obfuscated swear word since we consider such association as being "too far-fetched". We define it as $\epsilon = (\tilde{B_o} + \tilde{B_c})/2$.

As we are deviating from the "one size fits all" of single cost operation, it is appropriate that we consider that no single set of costs would be adequate in every situation. For example, obfuscations that change the length of a word can result in an word that is longer or shorter than the original. Those two situations require different preferences in edit operations. Thus multiple sets of operation costs may be considered and an heuristic can be used to select the more appropriate one.

### More edit operations

Just as the Damerau-Levenshtein algorithm can improve word recognition under certain circumstances [BM00] — not the ones we have at present, unfortunately — we decided to look at the possibility of extending the edit operations available to us based on our acquired knowledge.

Specifically, we noticed that character repetition was a common occurrence in the online forum we observed [LO14b]. Based on this, two new operations were created: *repetition* ($r$) doubles the previous character and its opposite, *unification* ($u$), removes a character if it follows another identical character. Since we are extending the Levenshtein operations, we will take this opportunity to further alter the substitution. The edit operation substitution had two possible costs, depending on the characters involved being equal, but from now on this behaviour will be separated into two different operations: *substitution* ($s$) and *keep* ($k$), the latter being the no-cost operation that is still a more specific version of the former. Table 8.3 compares both the specialised and general versions of all operations so far.

The extended operations roster is now as described in Equation 8.4. It should be said that the specialised operations have a more limited opportunity of being used and should take precedence over the generalised ones whenever possible. Ideally the more specific version should have a lower cost — even if they are more seldomly used — and an artificial way of lowering this may be useful, such as making this operation have extra weight during the frequency calculations. In Figure 8.8 we represent the map with all six edit operations present. The more specific operations are represented with lower cost than their more general siblings.

$$\begin{array}{rcl}
k((\sigma_1,\ldots,\sigma_n),\sigma_n) & = & (\sigma_1,\ldots,\sigma_n) \\
u((\sigma_1,\ldots,\sigma_{n-1},\sigma_n)) & = & (\sigma_1,\ldots,\sigma_{n-1}) \quad \text{iff } \sigma_{n-1}=\sigma_n \\
r((\sigma_1,\ldots,\sigma_n)) & = & (\sigma_1,\ldots,\sigma_n,\sigma_n) \\
d((\sigma_1,\ldots,\sigma_{n-1},\sigma_n)) & = & (\sigma_1,\ldots,\sigma_{n-1}) \\
i((\sigma_1,\ldots,\sigma_n),\sigma_m) & = & (\sigma_1,\ldots,\sigma_n,\sigma_m) \\
s((\sigma_1,\ldots,\sigma_{n-1},\sigma_n),\sigma_m) & = & (\sigma_1,\ldots,\sigma_{n-1},\sigma_m) = \\
 & = & i(r((\sigma_1,\ldots,\sigma_{n-1},\sigma_n)),\sigma_m) \\
 & & \sigma_i \in \Sigma, \forall i \in \mathbb{N}
\end{array} \tag{8.4}$$

But we can go even further, we could add more specific and specialised

Table 8.3: Comparison of corresponding specialised and general edit operations.

| Operations | Task |
|---|---|
| keep (*k*) | Replace a character with itself |
| substitution (*s*) | Replace a character with another (specified) character |
| unify (*u*) | Remove character if preceded by a similar one |
| delete (*d*) | Remove character |
| repeat (*r*) | Insert a character equal to the last one |
| insert (*i*) | Insert a character (specified) |

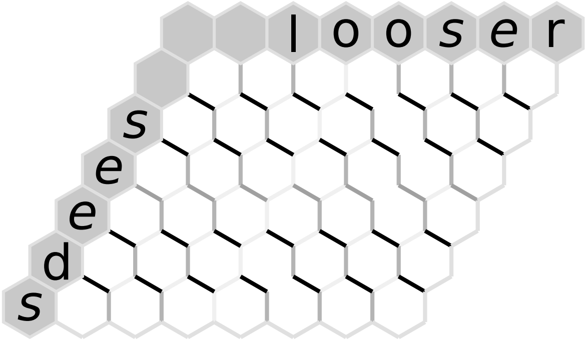


Figure 8.8: An update of Figure 8.7 that demonstrates the extended set of operations. The edges between the ‘ee’ and the ‘oo’ are shown lighter to represent lower-cost operations.

instructions. Many relevant candidates can be found in Table B.1, back on page 187. Of course, adding additional operations implies that we adjust the method we use to calculate the costs for the operations.

For clarification, our matching algorithm will continue to make use of the thee basic “classes” of operation: *insertion*, *deletion* and *substitution*. The difference now is that in each situation our algorithm will give preference to the more specialised operation available (as defined by the number of input values that can make it valid). For example, when removing a character it would first check if it was white space, then see if it was a punctuation signal, if it was a non-alphanumeric symbol, then verify if it constitutes a repetition and finally fall back into a general character deletion. That means that the graph in Figure8.8 will simply present a greater variety of costs.

## 8.3 Our evaluation

In this section we will describe the experiments we performed to evaluate the adequacy of our profanity recognition system. We will begin by recalling some information about the dataset we use and how we preprocessed the messages.

Our experimentation approach is quite straightforward. We compare the performance of our baseline method (described on Section 8.3.3) with three variants of our proposed deobfuscation methods (which we describe on the subsequent section). The details of the experiment are shared on Section 8.3.5 and on Section 8.3.6 we present our results as well as our analysis.

### 8.3.1 The dataset

The dataset that we used for our experiments was SAPO Desporto, which we described in detail back in Section 7.6. Here we will repeat some of the most relevant information.

SAPO Desporto is comprised of 2500 messages randomly selected from user comments posted at a Portuguese sports website [LO14b]. These messages, written in the native language, were annotated by hand, identifying all taboo words, including obfuscated cursing. 560 messages contained profanity, with 783 instances being recognised (words repeated in the same message are not counted if written in the same way), representing 109 different curse words.

Almost 70% of the profanity instances are obfuscated. This may be, in part, because of the automatic profanity filtering that the platform managers were enforcing. The obfuscation was not uniform, as 258 taboo word instances were spelled in an unique way, with an average of 3 spellings for each swear word.

### 8.3.2 Text preparation

We have previously (on Chapter 3) discussed how User-Generated Content presents its own set of pre-processing challenges, particularly on tokenization. Previous work [LO14a] confirmed the impact of pre-processing in recognising profanity, namely the tokenization and normalisation tasks.

As was said in that chapter, our tokenizer worked well for regular text tokenization, but proved to be a poor match when dealing with deobfuscation [LO14a]. This happens because the author of the message often "overrides" several characters that usually have very precise meanings, with particular problems arising from symbols considered *word dividers*, such as commas and full stops. But worse than that is the abuse of white space characters that are the most elemental word separators.

Training a model to handle this inconsistent usage is very difficult as semantics plays a very important role in resolving many situations. For example, in "s..." how do we know if we are looking at an ellipsis or an obfuscation? Context, and particularly semantic context, is essential.

The most common tokenization processes make use of regular expressions (as is the case with the Penn Treebank Tokenizer[2], for example). Such a solution is simple and can provide adequate results [LO14a], but these still present room for improvement; in particular, noisy texts can make them act unpredictably. Tweaking the regular expression can improve the recall for obfuscations, but at a disservice to regular words, which suggests that one single tokenization method is incapable of supporting such disparate interpretation of the symbols, and as a result, words/tokens. In fact, even for this single specialised task of deobfuscation, multiple approaches at tokenization are required to provide an accurate apprehension of the message as intended by the author (for example, ignoring word separators and attempting to recognise new words based in this way of interpreting the text). Thus, an accurate and complete preparation of the original message may depend on multiple iterations of different processes over the same source that are later combined into an unified view.

This line of work was consigned to the future since we did not consider it to be the dominant problem and was too specific, despite falling inline with some of our previous work. Our annotation already contained the relevant words correctly identified, meaning that we could avoid the shortcomings of imperfect tokenization in our experiments. Or, more precisely, we could avoid them when dealing with the cursing words. Our methods require a sample of non-swear words to fine-tune the parameters of the classifier, and these are not annotated. These words were taken from the messages that contained no profanity, to safeguard against the selection of part of a mistokenized taboo word. Tokenization was done by employing a simple regular expression (`\b\w+\b`), which was considered sufficient

[2] `http://www.cis.upenn.edu/~treebank/tokenizer.sed`, seen on 2014-12-20.

for our needs.

The normalisation process, which tries to reduce the different number of ways in which a word is written, was also shown to have a significant impact when deobfuscating words [LO14a]. We could say that deobfuscation is normalisation, and to avoid conflict or interference we only converted the messages to an all-lowercase version before working on it.

### 8.3.3 The baseline algorithm

The Levenshtein edit distance calculates the total cost for transforming the source string into a target string as a form of defining similarity. We can consider that a word in the text corresponds to a swear word if the (edit) distance we need to cover in order to transform it into said swear word is within a certain threshold. But defining a fixed threshold that works well for small and large words is difficult, since we already saw that obfuscation often changes the word length. A solution would be to define this threshold based on the length of the word we see [SAC12b], which provides better results.

Through experimentation we came to the value of $\epsilon = 0.5$ as providing the best results for this algorithm, that is, we allow up to half the characters on the word we see to be edited. All operation costs were maintained at 1 (with the natural exception of the *keep* operation).

### 8.3.4 The Levenshtein algorithm

We will take this opportunity to describe some implementation details that fall outside the general theory we have presented, and have a practical reason or existence.

**The number of paths** The greatest obstacle to running our algorithm is the sheer number of possible paths that can be created when converting a word into another (that is, $|P_i|$). Just to give a quick idea, if we have two words of size 2 (meaning that $|\phi_i| = |\omega_i| = 2$), then we have 6 different paths. Should both words be of length 4 — which are still small words — we are already dealing with 321 paths *for one word pair*. To get more representative values, in our data sample the average word length (from our full set of $\Sigma^*$) is 7 and the average swear word (in $\Phi$) is 6; such a pair would result in about 20,000 possible paths. The exact number of different paths can be calculated through Equation 8.5 where $i$ and $j$ refer to the length of the words.

$$T(i,j) = \begin{cases} 1 & \text{if } i = 0 \\ T(j,i) & \text{if } j < i \\ T(i-1,j) + T(i,j-1) + T(i-1,j-1) & \text{otherwise} \end{cases} \tag{8.5}$$

Each word in the text needs to be compared to each curse word from our swearing list (which is comprised of dozens of entries). We exclude source words that are part of a Portuguese dictionary and need only process each word once (and remember the result for later), but still it is apparent that it would be a struggle to deal with so many situations. We clearly need a search approach that is able to cull the search space.

We used the Dijkstra search algorithm [Dij59] to compute each path between the start and end node. This avoids looking through unhelpful paths (e. g. removing all letters and inserting the new ones with no regards to common substrings), and prevents the revisiting of nodes already processed. But how do we know what is a good or bad path? All edit operations look the same, we essentially have just an *insert*, a *delete* and a *substitute*, and different costs are not defined yet.

As we mentioned back on page 150, we will be selecting only the paths that emphasise each of the available operations. This means that we can cull our search space by pruning the paths that are falling below others in this regard. For example, if we take a look back at Figure 8.4 we can see that the two inner paths both contain the same operations: $1i + 5s$. That means that one of them is extraneous and will not be completed.

**Cost attribution** When converting edit operation frequencies to costs, the more specific operations should posses a lower cost than their more generic version (as stated on Section 8.2.2). To ensure this, we define the cost for each operation as follows: basic operations are accounted normally with a factor of 1. Specialised operation (other than *keep*, which always costs zero) are accounted as if they were seen more frequently than they actually were. The exact value is defined by $2S + 1G$, where $S$ represents the number of times the *specialised* operation was used and $G$ corresponds to the number of uses of the relevant *general* operation. These values were supported through experimentation and, since specialised operations are selected for use whenever possible, ensure that the algorithm looks at them favourably. Our operation costs are always rounded as integers.

**Default costs** It is possible that, during classification, our version of the Levenshtein algorithm tries to use an operation for which no cost has been attributed because it was never seen during the creation of the model being used. In that situation we use the largest cost that we did calculate.

**Two sets of costs** As was said in Section 8.2.2, some circumstances have requirements that can conflict with others. The most obvious situations, that we had presented as example, is that of obfuscation by shortening the word vs. obfuscation by lengthening the word — one can benefit from cheaper insertions while the other works better with cheaper deletions.

We split the "training" and "fitting" situations to deal with the shorting of words separately from the remaining cases. This means that, when pairing a word with a curse word, a different context will be used depending on how the former compares with the latter in length. With context we mean the contribution to a set of operation costs or the application of a set of costs. If more than one set of costs is applied, we compare all the candidates and elect the one associated with the lowest editing costs.

**The operations implemented** We implemented three versions of our proposed method, each one improving upon the previous one. This comes from Section 8.2.2.

The first version is similar to the baseline algorithm but allows for differing costs. The second one adds the operations $r$ and $u$, which derive from $i$ and $d$, respectively. The third one adds many operations, namely:

- Insert space,
- Insert punctuation,
- Delete space,
- Delete punctuation,
- Substitute a character with a diacritic by the same character without any,
- Substitute a character without a diacritic by the same character with a diacritic,
- Substitute a character with a diacritic by the same character with a different diacritic,
- Substitute a number with a letter,
- Substitute a letter with a number,
- Substitute a non-space character with a space character,
- Substitute a space character with a non-space character,
- Substitute a non-punctuation character with a punctuation character, and

- Substitute a punctuation character with a non-punctuation character.

Some of the operations were added only to provide symmetry to the operations and we do not expect them to be used on this work. The following operations were not specialised: *add letter*, *add number* and *change letter*, since we considered that the generic operations handle them adequately: $i$ and $s$ will only be introducing letters and the obfuscation operation $^{+}N$ is inexpressive in our corpus. The operations *word aggregation*, *complex alteration*, *phonetic-driven substitution* and *pun* were left out due to their complexity.

### 8.3.5 The experiment

The experiment was performed in a 10-fold cross-validation evaluation system, which was repeated 11 times (11 trials).

We employ the full set of 781 curse word instances, to which we add a random set of 781 non-curse words. As we mentioned before, the non-curse words were randomly sampled from messages containing no annotated swearing. The selection of these words, together with the shuffling determine the differences in results from one trial to the next. The words are all shuffled together before being partitioned; the baseline classifier ignored all but the test set, while our proposed classifiers used the data with the following approximate distribution for each cross-validation:

- Of the 781 annotated curse word instances
  - 90% is used for model creation, of which
    - 45% is used for the calculation of the costs candidates and
    - 45% is used for the costs selection and threshold calculation;
  - 10% is reserved for testing.
- Of the 781 non-curse words
  - 90% is used for the costs selection and threshold calculation and
  - 10% is reserved for testing.

The imbalanced situation that occurs on the cost selection is not as pronounced as we find "in the real world" (as we have seen in Section 7.3.1) and a variety of non-swear words helps the system avoid false positives. Repeating words were also accepted in the samples to reflect real word frequencies. Misclassification of a common word is undesirable since it would happen over and over, but with this arrangement common words would more likely be present during the model creation process, and this would reduce the likelihood of such error happening.

Our classifiers make use of a dictionary before attempting to classify a word. For such we employed the master dictionary for European Portuguese from GNU Aspell[3]. However, through a fault of ours, we used only a subset of the dictionary and not its full contents. Since this affected all classifiers equally (and not severely), and it illustrates a possible problem with dictionaries (they are never perfect) we decided it would not warrant a re-run of the entire experiment.

The label set that we used is comprised of all swear words that were present in our annotated corpus. This is different than using a list of swear words obtained from other means, as was the case of the similar works we previously mentioned.

### 8.3.6 Results and analysis

In our experiment we quantified the adequacy of the baseline and the four methods that we implemented using the standard Precision, Recall and F1 measurements. We consider a True Positive when the classifier accurately classifies a word from the source text as the swear word declared in the annotation. If the classifier proposes a non-swearing word as a disguised cursing, we consider it a False Positive. A True Negative happens when the classifier sees no significant relation between a non-taboo word and any taboo word in our swearing lexicon. Finally, a False Negative happens when our classifier ignores a swearing word.

We calculated the micro and macro averages for each 10-fold cross-validation experiment, as well as a weighted average that relates to the frequency of each label; these are standard measurements in classification evaluation. Micro averages for precision and for recall are always equal since a false positive for one class is a false negative for another. This is a consequence of us treating the absence of swearing as a regular class (we had, thus, 110 classes — which are too many to represent as a readable confusion matrix).

To summarise the performance across the multiple experiments we used the *median*, as we feel it provides a more accurate notion of what can be expected from the classifiers.

Since we are doing a multi-label classification experiment (and not applying a simpler binary classifier determining the mere presence of profanity), and the data that is being used is quite different from that employed in other studies, comparison with the work performed by others is very difficult.

[3] `http://aspell.net/`

Table 8.4: Median values of the performance measurement of the baseline classifier after all experiments were run. Each experiment calculated Precision, Recall and F1 for the 10-fold cross validation and presented the three averaging values.

| Average | Precision | Recall | F1 |
|---|---|---|---|
| micro | 0.90 | 0.90 | 0.90 |
| macro | 0.82 | 0.90 | 0.83 |
| weighted | 0.93 | 0.90 | 0.91 |

**The baseline**

We begin by presenting the results for the baseline classifier which is based on the standard Levenshtein algorithm. This method performed quite well when using proportional threshold, as this improves the results significantly when compared to a fixed edit distance (we can refer to a previous work of ours [LO14a] that evaluated the Levenshtein edit distance algorithm with a fixed threshold on the same dataset). The quite positive general results obtained are shown in Table 8.4, where we see the medians of all repetitions.

Sood, Antin and Churchill presented much lower results for a similar method [SAC12b] which, we can only assume, is the result of the swearing dictionary used, which was generic and obtained from unrelated sources.

Micro averages (calculated by the collection of all individual results) show that this classifier is fairly correct. Macro averages — the averages of the measurements taken per class — show a lower precision, meaning that *false positives* were more frequent than *false negatives*. The last line, weighted averages, is similar to macro averages but proportionally weights each class. Here we can see that precision increases significantly, which strongly suggests that the dominant class (i. e. not a swear word) is not the source of the *false negatives*. In fact, looking more detailed at the errors, we can see that the most common error was incorrectly assuming clean words were swearing (83 words on median), representing most of the errors committed. The converse situation — ignoring swear words that are present — is only seen a median of 42 times, while confusion about which swear word was actually seen occurred around 31 times.

We present some of the errors committed by the baseline classifier on Table 8.5. Some errors can be attributed to the dictionary that prevented a word from being properly evaluated (e. g. "carvalho"), but the dictionary we used also failed by not carrying some common words (e. g.' "como", "pontos" and "toda"). The dictionary was used as much for "word protection" as for reducing processing time, and this shortfall affects all classifiers equally, preserving their relative performance.

Table 8.5: Example of classification errors committed by the baseline classifier. One word was shortened to fit into this table.

| Word seen | Class expected | Class attributed |
|---|---|---|
| caaralhosssssss(...)sssssss | caralhos | — |
| carvalho | caralho | — |
| enculados | enrabados | encornados |
| f.d.p. | puta | — |
| m e r d a | merda | — |
| m*da | merda | foda |
| murcas | murcões | maricas |
| p*** | puta | — |
| putax | putas | puta |
| como | — | cago |
| piue | — | pila |
| pontos | — | bostas |
| toda | — | foda |

The table includes long and elaborate obfuscation that inserts too many characters for the baseline to handle (due to the threshold limit) and considers it as a non-curse word. In this table we can also see some "unfair" expectations: it is unlikely for "f.d.p." to be recognised employing our current methods just as "carvalho", being a dictionary word, needs a semantic analysis to resolve.

More importantly, we show some cases of "mistaken identity" where the classifier simply goes with another "similar" curse word (similar to its algorithm, not to humans) but ends being the wrong answer. This is the case with "m*da", where the classifier opted for the candidate solution *substitution-substitution-keep-keep* with total cost of 2 instead of the correct *keep-substitution-insert-keep* with the same total cost. The preference (and the error) was simply due to lexical order, but it does raise the question of how many correct classifications were made through luck in the "toss of the coin". This kind of situations fueled of our motivation to develop a model that better approaches the way people handle obfuscation. At the same time, ties in deobfuscation costs are not exclusive to the baseline and, despite being less common with our approaches, a better form of solving them could make a measurable difference.

We should also clarify that "piue" is not a Portuguese word (taboo or not), but is a corruption of a term that was used to mock based on the onomatopoeia of the singing of a bird ("piu" which the English write as "tweet").

### Our methods

We represent the results for the tests performed on the deobfuscation methods we have proposed as the difference in performance compared to the baseline. We label our proposed methods as *0*, *1* and *2* to refer to the three collections of edit operations we already described back in 8.2.2.

We plotted our results as Figure 8.9 where, at first view, we notice that across all forms of measurement the number of operations available correlate with the quality of the results obtained. That is to say that *method 0* shows a little degradation in performance compared with the baseline while *method 2* compares favourably with our reference method, with *method 1* exhibiting rather mixed results. We also notice that median recall shows very little in variation across all forms of measurement.

To help us better understand these results and to provide a more detailed picture of the classifiers, we plot the number of classification errors in Figure 8.10. These include the number of errors in recognising the swear words — which we further detail as the words that were disregarded and the words that were mistaken for other curses —, as well as the number of non-swear words that were classified in error as being profanities. Once more we use the results for the baseline as the basis of representation for the improvements and declinations of each of our classifiers. The rest of this section is devoted to a closer analysis of the results obtained from each classification method.

**Method 0** Analysing each of our proposals in more detail we can see that the simpler method — *method 0*, which is the most similar to the baseline — corrected many of the errors in the baseline classification. We highlighted these changes in classification on Table 8.6, that contains a sample of errors from this classifier. That was the result we expected with our modifications: solve the "exaggeration" obfuscations (such as in the first example).

However, with the improvements also came new errors in the classification (Table 8.6 has one example), and consequently the *method 0* classifier shows slightly worse results overall when compared with our reference algorithm, particularly on the precision measurement (as Figure 8.9 makes evident). Looking at Figure 8.10 we see that 20 more swear words were disregarded by the classifier, while 43 others were correctly taken as taboo but mistaken for other swear words (the baseline failed in 31 and this method exceeded that number by 12).

An examination of the classification revealed that many of the false positives were occurring with the popular swear word "cu". Many of its obfuscations were also 2 letters in length and the *substitution* was an edit operation too expensive to be considered, and for this reason a number of variations of his taboo word were overlooked.

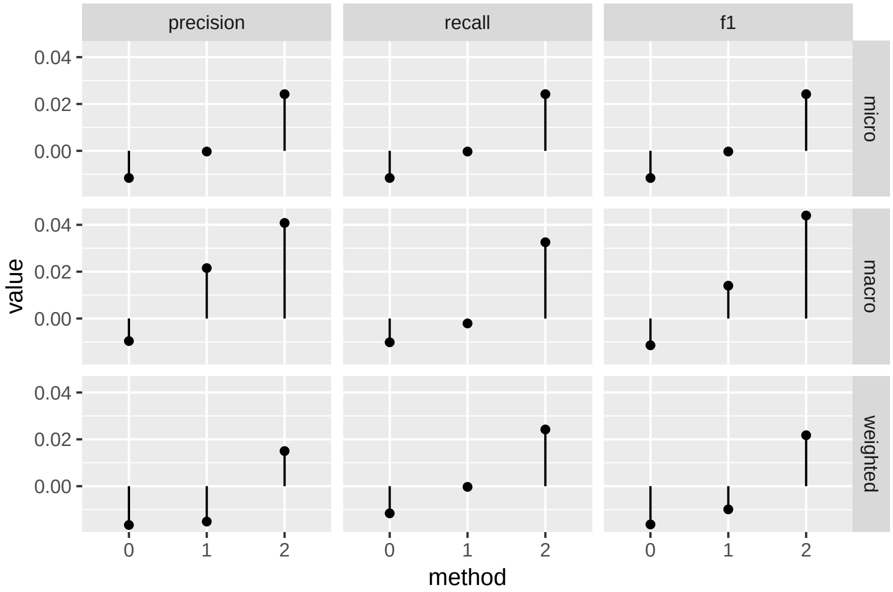


Figure 8.9: Results of our evaluation based on three standard measurements of classification performance. Three averaging methods were considered. The results presented are the median of the 11 trials and are represented as the difference from the baseline results presented on Table 8.4.

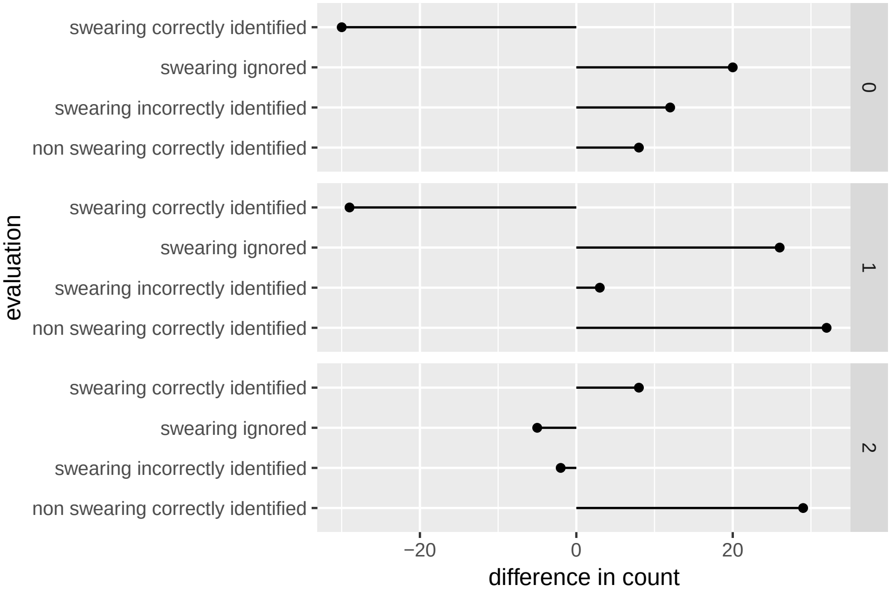


Figure 8.10: Detailed analysis of the classification errors for the three methods we developed. The results are presented as the differences from the baseline results in number of instances.

Table 8.6: How our *method 0* classifies the previous examples of classification errors committed by our baseline classifier (Table 8.5). One word was edited to fit in the table.

| Word seen | Class expected | Baseline | Method 0 |
|---|---|---|---|
| caaralhossss(…)sssssss | caralhos | — | caralhos |
| carvalho | caralho | — | — |
| enculados | enrabados | encornados | encornados |
| f.d.p. | puta | — | — |
| m e r d a | merda | — | merda |
| m*da | merda | foda | — |
| murcas | murcões | maricas | maricas |
| p*** | puta | — | — |
| putax | putas | puta | puta |
| como | — | cago | cago |
| piue | — | pila | — |
| pontos | — | bostas | bostas |
| toda | — | foda | foda |

**Method 1** The second method seems to be the most unremarkable of the three proposals. It scores very closely to the baseline in many tests (according to Figure 8.9), but it does present a very significant reduction in "imagined swear words" when compared with the previous method (based on Figure 8.10), that is, this method mistakenly identifies fewer non-cursing words as being cursing. This results in less false positives across many classes. In fact, we can say that this was the main gain over our *method 0*. However, this slight "bias" towards the "not a curse word" class also makes it ignore some actual cursing, resulting in significantly more false positives when evaluating this large class (and thus a lower weighted precision score compared to its unweighted score). We also noticed that the short taboo words were almost as problematic as they were for *method 0*.

The above interpretation can be substantiated with the examples of errors in Table 8.7. We see that most corrections to previous classification errors come from clearing words mistaken for cursing and dealing with repetitions (which was clearly the goal of this method). We also see how this new operation "backfiring" in one of the final cases shown. Still, the measurements show a positive progression and the final method will bring further improvements.

For the operation costs, the operations present in the previous method saw their costs increased, by a factor close to 2 (based on the same data). The threshold also raised, but slightly less, which means that situations that were previously close to this limit are no longer satisfied — that is, unless

Table 8.7: Comparison of errors committed by several classifiers in the same trial. Two of the words were edited to fit in the table.

| Word seen | Class expected | Baseline | Method 0 | Method 1 |
|---|---|---|---|---|
| campeão | — | cabrão | caralho | — |
| cobiçado | — | cagado | bico | — |
| cuuuu(...)uuuuu | cu | — | — | cu |
| có | — | cu | cu | — |
| enganas | — | enraba | enrabar | — |
| f0dasssseeeee | foda-se | — | — | foda-se |
| milhoes | — | colhões | colhões | — |
| raça | — | piça | piça | — |
| roupão | — | morcão | morcão | — |
| cabraao | cabrão | cabrão | cabrão | caralho |
| du@ta | puta | puta | puta | — |
| peeee(...)eeeerding | — | — | — | merda |
| pu t @ | puta | puta | puta | — |
| put@2 | puta | puta | puta | putas |

we are in a situation where a new, specialised operation can be used. As the *unify* operation is introduced, it may[4] cause the *delete* operation to rise in cost even further if it is seen being used. This happens because the more general operation is seen as being employed less often.

**Method 2** To conclude our analysis, *method 2* introduces new operations. We already saw in Figure 8.9 that it outperforms all other methods and the baseline across all forms of measurement we computed. From Figure 8.10 we can also understand that, while its recognition of non-swearing shows slightly less success than *method 1* did, it is the first of our methods to exceed the baseline in the complementary task of recognising swear words correctly.

Just as we saw an increase in the costs of the edit operations when going from *method 0* to *method 1*, so did they increased in the same fashion in *method 2*. In the same way, the value of the threshold increased but in a slightly smaller proportion. This allows for the introduction for new, cheaper and more specialised operations that are preferred in use.

As for the errors, Table 8.8 shows some errors (and a few successes) of this and other classifiers. We can see that, again, this is not a story made only of victories, and some words that have been correctly classified up until now by "simpler" classifiers are now being mislabelled. In particular, we can see that the small word "ku" still does not seem to be recognised.

[4] Remember that we create more than one set of costs.

Table 8.8: Comparison of errors committed by several classifiers in the same trial.

| Word seen | Class expected | Baseline | Method 0 | Method 1 | Method 2 |
|---|---|---|---|---|---|
| c o n @ | cona | — | — | — | cona |
| enculados | enrabados | encornados | encornados | encornados | enrabados |
| f.o.d.@ | foda | — | — | — | foda |
| p_u_t_@ | puta | — | — | — | puta |
| putax | putas | puta | puta | puta | putas |
| ca@brone | cabrão | cabrão | cabrão | cabrão | — |
| como | — | cago | cago | — | cocó |
| crlho | caralho | caralho | caralho | caralho | — |
| fodasse | foda-se | foda-se | foda-se | foda-se | fodesses |
| fodebol | fode | fode | fode | fode | fodendo |
| ku | cu | cu | — | — | — |
| murcóes | morcões | morcões | morcões | morcões | — |
| piue | — | pila | — | — | piça |

Table 8.9: Median number of words misclassified by *method 2* that were correctly handled by previous classifiers, and correct classifications performed by this classifier that were mishandled by the other classifiers.

| | Baseline, 0 and 1 | methods 0 and 1 | method 1 |
|---|---|---|---|
| Regressions | 4 | 9 | 11 |
| Progressions | 11 | 25 | 32 |

Short obfuscations are quite hard to decode since they allow few characters to work with. Should we increase their threshold then common words like "eu", "tu" or "ou" ("I", "you" or "or") could easily turn into false positives for taboo word "cu".

Despite these shortcomings, and based on the figures we have shown, we have already seen that *method 2* achieves more than any of the previous, thus such "slips" are more than made up with what it achieves above the rest. Table 8.9 paints exactly that picture, using numbers: for every word that our classifier stops getting right, it recognises 3 new word (median values).

For reference, the hard numbers for this classifier are disclosed on Table 8.10.

We also tried to determine the impact of the dictionary (Ω), overall. Running without this resource hurt the baseline method slightly more (by 2–5 percentage points) than it did our *method 2* (2 percentage points) across all averages. This was too much of a penalty for the two occurrences in

Table 8.10: Results for three measurements of our final classifier, *method 2*. The results were summarised using three different averages. The numbers shown are the median values obtained after the 11 repeats of the experiment.

| Average | Precision | Recall | F1 |
|---|---|---|---|
| micro | 0.92 | 0.92 | 0.92 |
| macro | 0.86 | 0.94 | 0.88 |
| weighted | 0.94 | 0.92 | 0.93 |

our corpus. However, we would expect that, should profanity censoring became tighter, this solution would become much more common.

**Comparison with other works** As we mentioned several times already, comparing our results is difficult for a number of reasons, namely different environments, different annotation types, different profanity dictionaries and different treatment of obfuscations. In addition to all that, we worked with different language and tried a more difficult challenge, since profanity erroneously classified as another profanity was considered a miss.

With all that, and with a hefty dose of *caveat emptor*, we can attempt to trace a quite subjective parallel of the numbers we presented in Section 7.5.3.

The work of Sood, Antin and Churchill [SAC12b] worked on Yahoo! Buzz and achieved the results summarised on Table 7.5 (on page 118). Their Levenshtein adaptation is quite similar to the one we used as the baseline, which makes us believe that the classification algorithm is not responsible for the discrepancy in numbers. In particular recall is severely low, which we believe could be due to 25/33 of the top swear words being obfuscated and tokenization not helping in these situations. In addition to all this, this corpus is more than twice the size of ours but the swearing was not annotated by hand.

Want et. al [WCTS14] worked on Twitter messages and, also trough a classifier based on the edit distance of Vladimir Levenshtein, achieved a near perfect precision, with 0.72 recall, combined in 0.83 F1 measure. This is a much better result, closer to ours. However, this corpus is smaller than ours (1000 tweets vs. 2500 forum messages), which may have reduced the variety of cussing and obfuscation.

Perhaps a better comparison could be made with an earlier work of ours [LO14a] where we used a tokenization and normalisation closer to the ones we believed to have been used in the studies we just mentioned. There we used again a method based on the Levenshtein edit distance, and obtained values of precision, recall and F1 of 0.96, 0.68 and 0.80, quite similar to the values of Want et. al. This is again quite similar to the baseline we used in the current experiments, and should extrapolate an idea of comparison, with due reservations.

But the important idea we would like to point out here is not one of numbers, but to present the importance of the pre-processing of messages when working with User-Generated Content, and how the introduction of noise can really affect higher-level processing of the text.

## 8.4 Summary and concluding thoughts

Taboo words may possibly be as old as civilisation. For example, over 4000 years ago old European tribes did not use the proper word for "bear", and referred to them by using the word for "brown". The "real name" of the animal was taboo, since they believed that mentioning its name would summon their presence. The English word "bear" actually derives from an ancient Germanic word for the colour of the fur of the animal, and their real name became disused [Vot]. This is the oldest example of obfuscation we are aware of.

Most taboo words are related to swearing, which, as we have discussed, is entrenched within our society in a special way. Despite being shunned by many, they have never really gone away — clearly they serve a purpose. These words lose their literal meaning and instead are used to *communicate* something at a higher level.

Offensiveness and acceptability dictate that, under certain situations, people may not express themselves as they would prefer. The obfuscation of taboo words is, thus, a way for people to preserve most of the manifestations they intended but avoiding some of the possible backlash and consequences that arise from the use of taboo words.

Profanity recognition systems have seen little attention in the scientific community, particularly in ours. Perhaps derived from the conflicting ideas or awkwardness of addressing such a lowbrow subject in a highbrow environment; or perhaps due to the deceptive apparent simplicity of the problem [SAC12a]. In addition — perhaps as a consequence of this lack of academic interest — profanity recognition systems seem to be built with the sole purpose of hiding or repressing, and not to expose or to understand. That part had been played for hundreds of years as humans apply social pressure and censure to put an end to foul language [McE06], with the success we all know and see. Could we realistically expect more from a machine?

Natural Language Processing systems have been created to overcome a number of hindrances, but purposeful introduction of noise, as far as we are aware of, was never one of it. With this work we are taking steps in that road.

Our goal was to study and improve automatic methods to *recognise obfuscated words*. The context of our application was that of swearing, due to the call for assistance of SAPO, and because profanity is the most common type of taboo words and the most frequent usage for obfuscation.

To this end we created a corpus dedicated to profanity identification that we annotated by hand and analysed in detail. We saw a correlation between swear words suppressed by filters and the variety of ways in which those words were written. We also determined the methods of obfuscation that were most frequently used and, based loosely on some of them, devised a series of classification methods to recognise the swear words disguised in messages.

Three classifiers were created, each one employing a different edit distance method based on the Levenshtein algorithm. Each of the methods has available a different set of specialised edit operations that can have different costs, with the more general operations — those present in the Levenshtein algorithm — being more expensive than their more specific variations. The costs are determined through the analysis of annotated examples, and here resides the core of our contribution.

Deobfuscation of a word we do not recognise is performed by searching for the most similar curse word using the edit distance. A threshold (computed at the same time as the costs of edit operations) limits the maximum dissimilarity we will consider.

Experimentation results indicate that the greater variety of edit operations available provide an improvement on the number of words that are correctly classified. Our most complex classifier showed about 5% macro average F1 improvement over our baseline classifier (derived very closely from the standard Levenshtein edit distance), while the micro and weighted averages indicate about 2% increase on this metric.

We concluded that our baseline is simpler, faster and provides quite adequate results; but it is also static and easier to fool (for example, by inserting many characters). Our methods do require an annotation, but this provides a path for adaptability to either different practices or to evolving environments, which is something that lists cannot provide in a flexible way. Online communication is very dynamic and subject to unpredictable changes, which are easier to account for through annotation than through modification of algorithms of hardcoded values. And since one of our classifiers does present a measurable improvement over the baseline, we do think that we achieved our goal of improving the state of the art in deobfuscation. That is not to say that everything went as expected, or that we have reached the end of this road.

We were surprised by the underperformance of our simpler method, given how similar it is to the baseline. We could have tweaked our methods to provide a more similar result — which could also reflect on our more complex derivatives —, but we preferred to maintain our approach simple and generic, and avoided focusing too much on our particular data. In the next section we will present a number of potential improvements and research directions that we would like to consider following of this work.

## 8.5 Future work

As usual, time was our main constrain, not lack of ideas. For this reason we feel there is still a significant ground to cover. In this section we describe the remaining steps in our plan that are still to be followed.

The first idea that we would like to evaluate is a *method 3* classifier. This would add a single new operation: substitute character $X$ with character $Y$. which would be able to address a number of situations which are common enough to have an impact. For example:

- Letters that sound alike — *c*, *k* and *q*, for example. This could solve our particular problem with "cu" / "ku", as in general small words need more careful choices during obfuscation, and soundalike characters are of greater importance in these situations.

- Common substitutions of non-letters, such as "0" and "o" or "@" and "a" that could be more precisely handled.

Similarly, we could also derive more specific operations from the *insert*. As we have seen, the vowels are usually the first letters to be deleted since they are fewer and due to their nature easier to add back. An operation *insert vowel X* could be added to handle those cases.

Still on the subject of more edit operations, and taking another step forward, it may be possible to automatically generate completely generic edit operations that would really learn the operations from the examples. By seeing many examples like the ones we just mentioned, it could learn such a mapping. This could replace all substitution operations — and similar adaptations could be made for *insertion* and *deletion*. Under-represented situations could be gathered in an "generic" version of the operation.

The main challenge arises from the dimension of our corpus; it may not be large enough to allow this precise operation to attain enough significance. The sparseness of the examples for each case may be insufficient to "learn" many of the useful substitutions. Consequently, adding more messages to the corpus would also be present on our road map.

We also noticed that our algorithm chooses between a number of possible costs for operations. Perhaps we could combine several suggested sets of costs into a single solution. We think there is only advantages in seeing more costs represented and in its current state some edit operations are omitted entirely from the final roster without necessity.

The threshold is currently calculated in a very simple way (the halfway point between two medians), which likely is non optimal. We would like to try other possibilities to generate better values for this parameter; one possibility is to use an iterative method that calculates the best value for $\epsilon$ by evaluating the number of mistakes that would be performed at each hypothetical point. We noticed it was usual to have some overlap between $\tilde{B_o}$ and $\tilde{B_c}$ (the costs for the obfuscations and the costs for the non-obfuscated words). Perhaps by better considering mistakes in a few rare situations we can improve the classification of more common situations.

Going a little further, improving and tailoring the specific pre-processing for the current task can go a long way in making or breaking an effort. In the tokenization side, methods of discovering "boken" words are required; perhaps doing a regular segregation into words, detecting areas of the message that "does not make sense" to be processed differently, and then merging the results into a final result. As for the normalisation, it is possible that our approach here was not the best one. We would like to evaluate the results of using the same method as Wang et. al [WCTS14], in which repetitions are blindly condensed. For example, "WWW" would be condensed into just "W" and "butt" would be turned into "but", but the principle may have merit. Not only would it make deobfuscations word-length independent as it could allow the edit distances to less dominated by the *unification* operation. However, more attention would be required since the profanity classifier could be reporting on a word that is not in the text. An entire new pre-processing procedure may just be the best solution.

As we stated before, per-user customisation could also help with resolving some situations, as the system could be tailored to resolve the preferences of a single person; but the difficulty of amassing the required volume of examples from a single user to make it worthwhile should be admitted.

Finally, people do not usually deobfuscate swearwords in isolation; they look at the chaining of the words and when needed they use their experience to help them complete the pattern. It is for this reason that obfuscations such a "son of a *****" work: it is such a normative expression that people already expect it, and that buys a greater degree of freedom to the obfuscating author. To acquire such information we generated n-grams from Twitter messages and, while we learned that it cannot tell us what swear word to expect due to the variety of it (save for very few expressions as the one above), we can use it to learn what swear words we have seen (even if infrequently) in such context. This information could solve ties between curse words and to help correct the number and gender of curse words presented by our classification.

# Chapter 9

# Concluding remarks

The structure of the present work prescribes a conclusion and future work in the work chapters. This section is thus used to provide a general overview of the work and the future direction we would like to take.

Microblogs brought us a new form of sporadic communication, based on short and immediate messages that can reach millions of people across the globe in an instant. The barrier to participation is fairly low since the (artificial) limitation placed on the message length helps to blur the distinction between good writers and bad writers, people with a message to say and people who simply chat, and is, in this way, quite inclusive. The value of microblogs is, in a large amount, centered in the personal world views that are shared by their users. Unfortunately it is not easy to get to them (automatically).

The language used in microblogs is not the "conventional" language used in the traditional written word. It is constrained in some ways and exceeding liberal in others, it is strongly influenced by oral communication, it tries to express emotion through new means, users try to set themselves apart from each other through several creative methods... Adding to this we have the mobile devices with their limited keyboards as well as the usual diminished context present in each message — occasionally even humans feel challenged when reading microblog posts.

The pre-processing and normalization phases of messages are of paramount importance. They "translate" the microblog messages to a more common form, and facilitate the use of higher-level text processing tools (such as part of speech tagging, sentiment analysis and opinion mining).

In the present work we focused in a number of specific problems that are related to the pre-processing of User-Generated Content, with a strong emphasis in microblogs. These problems were practically motivated by our participation in research projects that were faced with these very problems when processing data from such sources.

Our research has shown how the process of *tokenization* can be quite complex on microblogs, not so much due to the special content that exist (e. g., URLs and hashtags), but because users usurp the long-established meaning of certain symbols, using them to create little faces (smileys) or new codes (like "xoxo" to mean "hugs and kisses") or abbreviations (e. g. "/." for the website "Slashdot").

Faced with the very difficult task of generating and maintaining a rules-based system to tokenize such high-entropy messages, we opted to employ a machine learning method to infer correct rules of tokenization and redefined the entire process as a classification problem (in Chapter 3). This had two major benefits: this system provides better performance (exceeding the 0.95 $F1$ value) than the regular-expression tokenizer we created explicitly for microblogs, but it also reduced the cost of maintenance by improving much faster than a rules-based system: our tests showed how providing more relevant examples is quite fast, simple, effective and safe (adding new rules may have unforeseen consequences).

The task at hand defines what can be defined as a *token*. In the common case of regular and generic "text processing" — semantic analysis for example, where tokens represent workds, smileys, URLs. . . — our tokenizer works very well. In other, lower-level situations (as is the case of *deobfuscation* that we first approach in Chapter 7), the problem is turned upside-down: *noise* is not something to be tolerable, ignored or discarded, *noise is the data* and was purposefully made difficult to process. Everything else is, at most, a secondary consideration.

In this context obfuscation is the act of hiding text in a way that it is harder to understand. Its most common use is disguising swearing which is the setting in which we developed our work. As the name says, "deobfuscation" is the act of exposing the original text.

We worked with messages taken from a popular sports website that we annotated into (we believe) the first free-access corpus for the study of obfuscation. Based on our observations we are lead to believe that, when a filter is introduces to curb cursing, a significant number of users prefer to disguise their swearing rather than changing their language. We also documented preferences in the methods used to disguise their taboo words, some of which we cannot realistically expect to be handled by our current state of the art computing.

With this knowledge we proceeded to design a method, based on extensions to the Levenshtein Edit Distance that would be able to better recognize the true, hidden word. This proved to be a very difficult (adversarial) problem and we managed to obtain only a small improvement on the baseline — but still exceeding the 90% F1 measurement in classification. This work is described in Chapter 8 where we also describe some very specialized tools that are required to better address it.

A very different type of problem we approached was related to *user classification*, where we try to determine something about users based only on their writings. This prompted us to delve into the (language agnostic) *writing style* employed by the users. We tested the relevancy of our *stylistic features* by trying to assign authorship of several messages based on past samples, using an automatic classifier. This challenge is described in Chapter 4, where we reveal to achieve accuracy results exceeding the 0.6 maximum $F1$ values with 3 candidates. Such result is comfortably above the choice by pure chance (1/3). The most revealing characteristics were the *marks of emotion*, that include the way users express laughter and write smileys, for example.

When it came to recognize automatic posting systems (also known as *bots*), we employed another classification system where the stylistic features proved very useful (reaching about 0.9 accuracy by themselves in our experiments). This work is presented in Chapter 5, where we also explore other revealing features that helped us boost the classification accuracy to 0.97.

We also worked on trying to determine the nationality of a user based on the variant of the language they use. We explored this problem with the American and European variants of Portuguese (in Chapter 6). It was surprising to us that the stylistic features were not as helpful this time. That probably meant that there was no "nation-wide" style of writing and that the dissemination of features such as the use of smileys and the indication of laughter are culturally disseminated. To solve this problem we saw a great contribution from our lists of words and expressions that are used in both sides of the Atlantic. We were able to reach 0.95 accuracy when scanning 100 messages from each user, which was higher than the n-gram approach that is commonly used for identifying languages.

Many ideas and directions are described at the end of each work chapter. But if we had to choose the one where the greater impact would be felt, it would be in the apparently basic tokenization process. We are convinced that there is a significant ground to cover towards a greater noise resilience, and a multi-stage approach would provide not only better accuracy but also more information for subsequent tools in the pipeline.

# Appendix A

# Nationality hint words

Table A.1: Words related to the Portuguese (European) variant of the Portuguese Language

| Distrits | Provinces |
|---|---|
| Aveiro | Alentejo |
| Beja | Algarve |
| Braga | Alto Alentejo |
| Braganca | Alto Douro |
| Bragança | Baixo Alentejo |
| Castelo Branco | Beira |
| Coimbra | Beira Alta |
| Evora | Beira Baixa |
| Évora | Beira Litoral |
| Faro | Douro |
| Guarda | Douro Litoral |
| Leiria | Estremadura |
| Lisboa | Minho |
| Lisbon | Ribatejo |
| Portalegre | Tras-os-Montes |
| Porto | Trás-os-Montes |
| Santarem | Trás-os-Montes e Alto Douro |
| Santarém | |
| Setubal | |
| Setúbal | |
| Viana do Castelo | |
| Vila Real | |
| Viseu | |

| Football | Companies |
|---|---|
| Benfica | CP |
| FCP | Clix |
| Futebol Clube do Porto | EDP |
| Porto | Optimus |
| Porto | PT |
| SCP | Phone-ix |
| SLB | Portugal Telecom |
| Sporting | Rede4 |
| lagarto | SAPO |
| lagartos | TAP |
| lampiao | TMN |
| lampioes | Uzo |
| lampião | Vodafone |
| lampiões | Yorn |
| tripeiro | |
| tripeiros | |

| Personalities | Vocabulary |
|---|---|
| Alberto Martins | Euro |
| Ana Jorge | Nestum |
| Antonio Costa | a gente |
| António Costa | abarrotar |
| BE | acto |
| Bloco de Esquerda | agreste |
| CDS | algures |
| CDS-PP | amo-a |
| CDU | amo-o |
| Cavaco Silva | amo-te |
| Comunista | apetece |
| Comunistas | apetecer |
| Democrata | autocarro |
| Democrats | biliao |
| Durao Barroso | bilião |
| Durão Barroso | briosa |
| Fernando Pinto Monteiro | brioso |
| Francisco Assis | bue |
| Francisco Louca | bué |
| Isabel Alcada | camiao |
| Isabel Alçada | camião |
| Jorge Lacao | casa-de-banho |
| Jorge Lacão | catraio |

Jose Seguro
Jose Viegas
José Seguro
José Viegas
João Proenca
João Proença
Loucã
Louça
Louçã
Luis Filipe Vieira
Luís Filipe Vieira
Mario Nogueira
Miguel Macedo
Miguel Relvas
Mourinho
Mário Nogueira
PP
PS
PSD
Partido Comunista
Partido Socialista
Passos Coelho
Paulo Macedo
Paulo Portas
Pinto da Costa
Portas
Santos Silva
Sociais Democratas
Social Democrata
Socialista
Socialistas
Socrates
Sócrates
Teixeira dos Santos
Vitor Gaspar
Vítor Gaspar

chavala
chavalo
comboio
cá
electrónica
facto
fixe
gaiato
gajo
ginasio
ginásio
gira
giras
giro
giros
golo
invicta
lavar os dentes
lx
mais pequeno
mb
melga
miuda
miudas
miúda
miúdas
multibanco
pa
parvo
parvoice
parvoíce
petiscar
portugal
portugues
portuguesa
portuguesas
portugueses
português
presidente da câmara
pá
sacas
se calhar
sesta
tabaco

tasca
telemovel
telemóvel
toda a gente
treta
triliao
trilião
tu
vós
ya

Table A.2: Words related to the Brazilian (American) variant of the Portuguese Language

| States | Cities |
|---|---|
| Acre | Belo Horizonte |
| Alagoas | Brasilia |
| Amapá | Brasília |
| Amazonas | Curutiba |
| Bahia | Forteleza |
| Ceará | Manaus |
| Distrito Federal | Porto Alegre |
| Espirito Santo | Recife |
| Espírito Santo | Rio |
| Goiás | Rio de Janeiro |
| Maranhão | Salvador |
| Mato Grosso | Sao Paulo |
| Mato Grosso do Sul | São Paulo |
| Minas Gerais | |
| Paraiba | |
| Parana | |
| Paraná | |
| Paraíba | |
| Pará | |
| Pernambuco | |
| Piaui | |
| Piauí | |
| Rio Grande do Norte | |
| Rio Grande do Sul | |
| Rio de Janeiro | |
| Rondonia | |
| Rondônia | |

| | |
|---|---|
| Roraima | |
| Santa Catarina | |
| Sergipe | |
| São Paulo | |
| Tocantins | |
| Football | Companies |
| Atletico | Band |
| Atletico Paranaense | Bandeirantes |
| Atlético Mineiro | Claro |
| Atlético Paranaense | Globo |
| Bambi | InfrAero |
| Bota Fogo | Oi |
| Corinthians | Petrobras |
| Cruzeiro | Petrobrás |
| Curintia | Record |
| Flamengo | Rede Globo |
| Fluminense | Rede TV |
| Gremio | SBT |
| Grêmio | TV Cultura |
| Internacional | Tam |
| Mengao | Tim |
| Mengão | Vale do Rio Doce |
| Palmeiras | Vivo |
| Paranaense | Voe Gol |
| Santos | |
| São Paulo | |
| Urubu | |
| Urubus | |
| Vasco | |
| Vasco da Gama | |
| Personalities | Vocabulary |
| Adriana Galisteu | BBB |
| Adriana Lima | academia |
| Alcione Nazaré | acesse |
| Ayrton Senna | alo |
| Caetano veloso | alô |
| Cláudia Leite | amo voce |
| DEM | amo você |
| Didi | apavorado |
| Dilma | avacalhar |
| Eliana | aí ela |
| Fafa | aí ele |

Fafá
Faustao
Faustão
Fenomeno
Fenómeno
Galvao Bueno
Galvão Bueno
Ganso
Gisele
Gugu
Ivete Sangalo
Jo Soares
Jô Soares
Lima Duarte
Lucas
Luis Fabiano
Lula
Luís Fabiano
Neymar
PC do B
PDT
PMDB
PPS
PSDB
Panicat
Rodrigo Santoro
Rogerio Ceni
Ronaldinho
Tiririca
Xuxa

aí nós
aí você
babado
bacana
bala
banheiro
barramento
bicha
bilhao
bilhoes
bilhão
bilhões
bobao
bobo
bobão
bombom
bombril
botar
botaram
botou
brasil
brasileira
brasileiras
brasileiro
brasileiros
brega
brotar
cachaca
cachaça
cachorra
cachorro
cade
cadê
caixa electronica
caixa electrônica
cala a boca
caminhao
caminhão
cara
celular
chiclete
copa
credo
câncer

demais
desencana
desetressa
direito
doida
doidao
doido
doidão
durmir
econômico
eita
electronica
electrônica
eletrônica
eletrônico
em meu
em minha
enturmar
equipe
escovar os dentes
esporte
é negócio
fala sério
festão
fofoca
galera
ganhador
garota
garoto
gaucho
gaúcho
gente
gol
gorducha
gringo
jeito
legal
libertadores
linda
lindo
machuca
machucada
machucado
machucou

machuquei
mais tem
malucona
mamae
mamãe
mané
mascada
mascado
mascando
mascar
metrô
milha
milhas
moca
moco
molenga
moleque
moleza
mouse
moça
moço
mulherada
na raça
ne
negao
negrao
negrão
negão
nessa
nesse
num
num fosse
num pode
né
onibus
ónibus
orkut
ô louco
papai
pegar
pelada
peladinha
picanha
pisar na bola

pisou na bola
planejar
planeje
planejei
planejo
planejou
poxa
pra
presente
pro
puxa
puxa saco
puxa vida
qi
que saco
quero ver
rola
rolar
rolou
samba
sambar
sambei
saque
sei não
sertao
sertão
show
so tem
soneca
sossega o facho
só tem
te amo
time
to
todo mundo
tolo
tomar cafe
tomar café
tou nem aí
trem
trilhao
trilhoes
trilhão
trilhões

| |
|---|
| tô |
| ué |
| vagabunda |
| vagabundo |
| vai na |
| vai no |
| vai pro |
| vc |
| vcs |
| vei |
| vem nao |
| vem não |
| vestibular |
| viado |
| viu |
| voce |
| voces |
| você |
| vocês |
| vovó |
| véi |
| xoxar |
| zoar |

# Appendix B

# Swear words

Table B.1: Methods used by the authors to obfuscate swear words in our sample of the SAPO Desporto dataset.

| Symbol | Name | Description |
|---|---|---|
| $^{-}Ac$ | Accent removed | A diacritical mark is removed from the word. Examples: "cabr**ã**o" → "cabr**a**o", "pi**ç**o" → "pi**c**o". |
| $^{-}C$ | Character removed | One or more letters and/or symbols are removed from the word. Such as "merda" → "mrda" |
| $^{+}Ac$ | Accent added | A diacritical mark is added where it is not supposed to exist. This alteration seldomly has any phonetic impact on the words. For example: "c**o**có" → "c**ó**có". |
| $^{+}L$ | Letter added | An extra letter is added to the word, but is not a repetition of the preceding letter. Example: "paneleiro" → "pan**d**eleiro". |
| $^{+}N$ | Number added | A seemingly random number is added to the word. Single occurrence: "puta" → "put@**2**". |
| $^{+}P$ | Punctuation added | A punctuation sign, or similar character, is inserted into the word. We noticed that these characters are often used as separators since they are easily distinguished from letters. An author may choose to use more than one such character in a word. Example: "fodam" → "f-o-d-a-m". |
| $^{+}S$ | Symbol added | A symbol not from our punctuation set is inserted in a word. Example: "foder" → "fu**£**der". |

| Symbol | Name | Description |
|---|---|---|
| $^{+}Sp$ | Space added | A space is used to break the word into two or more segments. As in "cornos" → "co rnos", "puta" → "p u t a". |
| $^{=}Ac$ | Change accent | One letter with an accent is replaced by the same letter bearing a different accent. No pure example appears in our sample but it can be observed in "morc**õ**es" → "murc**ó**es". |
| $^{=}L$ | Letter substituted | One letter is replaced by one other letter. Usually this change does not alter the pronunciation of the word. |
| $^{=}N$ | Number as a letter | A number takes the place of a letter. For example, "3" replaces an "e", "4" stands for an "a", and "0" is used as an "o". Often the digit that is introduced shares a graphical resemblance with the letter it replaces. Example: "f**o**da" → "f**0**da". |
| $^{=}P$ | Punctuation as a letter | One of the characters of our punctuation set are used as a placeholder for one or more letters. Example: "puta" → "p**...**". |
| $^{=}S$ | Symbol as a letter | A symbol from outside our punctuation set is used as a letter. Example: "m**e**rda" → "m**€**rda". |
| $Ag$ | Word aggregation | Two words are combined into just one, but not always by simple concatenation, like a portmanteau. For example, "americuzinho" combining "Américo" and "cuzinho" ("cu" and "co" sounding similar in this case). |
| $Cp$ | Complex alteration | We use this classification to encompass all forms of obfuscation that are too complex to be described with the other methods. A common occurrence is "fdp", that are the initials for "son of a bitch" ("sob") in Portuguese. |
| $Ph$ | Phonetic-driven substitution | The author changes the word in ways that make it sound the same when reading, but went beyond a simple letter substitution. Example: "foda-se" → "foda**sss**e". |
| $Pun$ | Pun | The author relies on the context to explain the obfuscation. One example is "caralho" → "carvalho" (which is Portuguese for "oak tree"). |

| Symbol | Name | Description |
|---|---|---|
| *R* | Repetition | One letter is repeated. As an example we have "merda" → "merd**dddddddd**a". |

Table B.2: All 109 profanities put into 40 groups, showing the number of different ways each one was written. Words filtered by SAPO are in bold. The instances column accounts the popularity of said group.

| Instances | Number of variants and swear words |
|---|---|
| 5 | 1 badalhoca, 1 badalhoco, 1 abadalhocado |
| 4 | 1 bastardos |
| 9 | 1 bico, 1 bicos |
| 39 | 3 bosta, 2 bostas |
| 7 | 3 **broche**, 1 **broches**, 2 brochista |
| 47 | 14 **cabrão**, 5 **cabrões** |
| 20 | 1 cagado, 1 cagadeira, 4 **cagalhão**, 2 **cagalhões**, 1 caganeira, 1 cagarolas, 3 cago, 1 caguem |
| 36 | 18 **caralho**, 1 **caralhinhos**, 2 **caralhos** |
| 1 | 1 chulecos |
| 2 | 2 cocó |
| 3 | 3 colhões |
| 6 | 5 **cona**, 1 conas |
| 191 | 9 **corno**, 8 cornos, 1 cornudas, 3 **cornudo**, 1 cornudos, 1 encornada, 1 encornadinhos, 2 encornado, 1 encornador, 2 encornados, 1 encornar, 1 encornei |
| 71 | 11 cu, 3 **cuzinho** |
| 9 | 1 enraba, 1 enrabada, 1 enrabado, 2 enrabados, 2 **enrabar**, 1 enrabá-lo |
| 6 | 3 **esporra**, 1 esporrada |
| 47 | 8 **foda**, 3 fodam, 9 foda-se, 4 **fode**, 1 fodei-vos, 1 fodem, 1 fodendo, 5 **foder**, 1 fodesses, 1 fodeu, 2 fodida, 2 fodido, 2 fodidos, 1 fodo |
| 2 | 1 mamões, 1 **mamadas** |
| 5 | 2 maricas, 1 mariconço |
| 1 | 1 masturbar-se |
| 87 | 40 **merda**, 1 merditas, 2 **merdoso**, 1 bardamerda |
| 1 | 1 mijo |
| 29 | 6 morcão, 1 morcãozada, 5 morcões, 1 morcona |
| 2 | 1 pachacha, 1 pachachinha |
| 8 | 7 **panasca**, 1 panascos |
| 13 | 9 **paneleiro**, 1 paneleirice, 2 **paneleiros**, 1 paneleirote |

| Instances | Number of variants and swear words |
|---:|---|
| 2 | 1 panisgas |
| 5 | 5 **peida** |
| 4 | 2 **piça**, 1 piças |
| 2 | 2 **picha** |
| 4 | 2 **pila**, 1 pilas |
| 5 | 3 piroca, 2 pirocas |
| 2 | 1 pisso |
| 12 | 1 pizelo, 1 pizelos |
| 9 | 1 porcalhota |
| 1 | 1 punhetas |
| 73 | 35 **puta**, 5 **putas**, 1 putéfia, 1 putinha |
| 9 | 2 rabeta, 1 rabetas |
| 1 | 1 rameira |
| 3 | 3 **tomates** |

# Bibliography

[AC08] Ahmed Abbasi and Hsinchun Chen. Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. *ACM Trans. Inf. Syst.*, 26(2):1–29, 2008.

[ACDK08] Rema Ananthanarayanan, Vijil Chenthamarakshan, Prasad M Deshpande, and Raghuram Krishnapuram. Rule based synonyms for entity extraction from noisy text. In *Proceedings of the second workshop on Analytics for noisy unstructured text data*, AND '08, page 31–38, New York, NY, USA, 2008. ACM.

[Alm14] José João Almeida. *Dicionário aberto de calão e expressões idiomáticas*, October 2014. (Retrieved on 2017-08-07).

[ANSR08] Sreangsu Acharyya, Sumit Negi, L. V. Subramaniam, and Shourya Roy. Unsupervised learning of multilingual short message service (SMS) dialect from noisy examples. In *Proceedings of the second workshop on Analytics for noisy unstructured text data*, AND '08, page 67–74, New York, NY, USA, 2008. ACM.

[BDF$^+$13] Kalina Bontcheva, L Derczynski, A Funk, M.A. Greenwood, Diana Maynard, and Niraj Aswani. TwitIE: An open-source information extraction pipeline for microblog text. In *International Conference Recent Advances in Natural Language Processing, RANLP*, page 83–90, 01 2013.

[Ber17] Leonid Bershidsky. Why I didn't laugh at Trump's 'covfefe' tweet: Bloomberg View. `http://www.oregonlive.com/opinion/index.ssf/2017/06/why_i_didnt_laugh_at_trumps_co.html`, jun 2017.

[BFD$^+$10] Sriram Bharath, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. Short text classification in twitter to improve information filtering. In *Proceeding of the*

*33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, page 841–842, New York, NY, USA, 2010. ACM.

[BI14] Arturs Backurs and Piotr Indyk. Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false). *CoRR*, abs/1412.0348, 2014.

[BM00] Eric Brill and Robert C Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, page 286–293, Morristown, NJ, USA, 2000. Association for Computational Linguistics, Association for Computational Linguistics.

[BMRA10] Fabrício Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgílio Almeida. Detecting spammers on Twitter. In *Proceedings of the 7th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*, 2010.

[Boi09] Leonardo Boiko. How to count to 140: or, characters vs. bytes vs. entities, third strike. `http://www.mail-archive.com/twitter-development-talk@googlegroups.com/msg06972.html`, May 2009.

[Bra70] Fernand Brandel. *Civiliazação material e capitalismo*, volume 1. Edições Cosmos, 1970.

[Bul09] Susanne Bullo. Microblog Battle - Twitter, Jaiku and Identi.ca. `https://web.archive.org/web/20110127203656/http://www.suite101.com/content/microblog-battle-twitter-jaiku-and-identica-a170508`, November 2009. (Retrieved on 2011-01-23).

[Bur10] Rick Burnes. When Do Most People Tweet? At the End of the Week. `http://blog.hubspot.com/blog/tabid/6307/bid/5500/When-Do-Most-People-Tweet-At-the-End-of-the-Week.aspx`, January 2010.

[Cad] Sean Cadhain. Teen textuality and the txt flirt. `http://www.txt2nite.com/teen-textuality-and-the-txt-flirt.html`. (Retrieved on 2017-04-22).

[CCL10] Zhiyuan Cheng, James Caverlee, and Kyumin Lee. You are where you tweet: a content-based approach to geo-locating twitter users. In *CIKM*, page 759–768. ACM, 2010.

[CDPS09] Noah Constant, Christopher Davis, Christopher Potts, and Florian Schwarz. The pragmatics of expressive content: Evidence from large corpora. *Sprache und Datenverarbeitung: International Journal for Language Data Processing*, 33(1-2):5–21, 2009.

[CGWJ10] Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. Who is tweeting on Twitter: human, bot, or cyborg? In Carrie Gates, Michael Franz, and John P. McDermott, editors, *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, page 21–30, New York, NY, USA, 2010. ACM.

[Che] Marc Cheong. 'What are you Tweeting about?': A survey of Trending Topics within Twitter. `http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=C213B0C404F694DB7F76EAFC0EB4E59B?doi=10.1.1.158.4679&rep=rep1&type=pdf`. (Retrieved on 2018-01-04).

[Che18] Mike Cherney. Lawyers Faced With Emojis and Emoticons Are All _("/)_/. `https://www.wsj.com/articles/lawyers-faced-with-emojis\-and-emoticons-are-all-\1517243950`, jan 2018. Via `https://9to5mac.com/2018/01/30/emoji-case-law/` (Retrieved on 2018-02-01).

[CHFT07] Richard J. Crisp, Sarah Heuston, Matthew J. Farr, and Rhiannon N. Turner. Seeing Red or Feeling Blue: Differentiated Intergroup Emotions and Ingroup Identification in Soccer Fans. *Group Processes Intergroup Relations*, 10(1):9–26, February 2007.

[Chi11] Oliver Chiang. Twitter Hits Nearly 200M Accounts, 110M Tweets Per Day, Focuses On Global Expansion. `http://blogs.forbes.com/oliverchiang/2011/01/19/twitter-hits-nearly-200m-users-110m-tweets-per-day\-focuses-on-global-expansion/`, January 2011.

[CL01] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[CL09] Marc Cheong and Vincent Lee. Integrating web-based intelligence retrieval and decision-making from the twitter trends knowledge base. In *SWSM '09: Proceeding of the 2nd ACM workshop on Social web search and mining*, page 1–8, New York, NY, USA, 2009. ACM.

[CMP11] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, WWW '11, page 675–684, New York, NY, USA, 2011. ACM.

[CNN$^+$10] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi. Short and Tweet: Experiments on Recommending Content from Information Streams. In *ACM Conference on Human Factors in Computing*, Atlanta, GA, 04/10/2010 2010. Association for Computing Machinery, Association for Computing Machinery.

[Coh61] Marcel Cohen. *A escrita*. Colecção Saber. Publicações Europa-América, Rua das Flores, 45, Sept 1961.

[Coh98] Elliot D. Cohen. Offensive message interceptor for computers, August 1998.

[con18a] Wikipedia contributors. Microblog in China. `https://en.wikipedia.org/wiki/Microblogging_in_China`, February 2018. (Retrieved on 2018-02-05).

[con18b] Wikipedia contributors. Seven dirty words — Wikipedia, The Free Encyclopedia. `https://en.wikipedia.org/w/index.php?title=Seven_dirty_words&oldid=819510861`, 2018. (Retrieved on 2018-01-10).

[con18c] Wikipedia contributors. Tencent QQ. `https://en.wikipedia.org/wiki/Tencent_QQ`, February 2018. (Retrieved on 2018-02-05).

[CT94] William B. Cavnar and John M. Trenkle. N-Gram-Based Text Categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, page 161–175, 1994.

[Cul10] Aron Culott. Detecting influenza outbreaks by analyzing Twitter messages. In *1st Workshop on Social Media Analytics (SOMA '10)*, July 2010.

[CWT13] Simon Carter, Wouter Weerkamp, and Manos Tsagkias. Microblog Language Identification: Overcoming the Limitations of Short, Unedited and Idiomatic Text. *Language Resources and Evaluation Journal*, 2013.

[Dam64] Fred J. Damerau. A Technique for Computer Detection and Correction of Spelling Errors. *Commun. ACM*, 7(3):171–176, March 1964.

[DH09] Lipika Dey and Sk. Mirajul Haque. Opinion mining from noisy text data. *International Journal on Document Analysis and Recognition*, 12:205–226, 2009.

[Dij59] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[DMAB13] Leon Derczynski, Diana Maynard, Niraj Aswani, and Kalina Bontcheva. Microblog-genre Noise and Impact on Semantic Annotation Accuracy. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*, HT '13, page 21–30, New York, NY, USA, 2013. ACM.

[dSL06] Joaquim Ferreira da Silva and Gabriel Pereira Lopes. Identification of Document Language is not yet a completely solved problem. In Lotfi A. Zadeh and Stephen Grossberg, editors, *Proceedings of the International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA-IAWTIC'06)*, page 212–219, Los Alamitos, CA, USA, November 2006. IEEE Computer Society.

[dVACM01] O. de Vel, A. Anderson, M. Corney, and G. Mohay. Mining e-mail content for author identification forensics. *SIGMOD Rec.*, 30(4):55–64, 2001.

[Eag94] Robert Eagleson. Forensic analysis of personal written texts: a case study. In John Gibbons, editor, *Forensic Linguistics: An Introduction to Language in the Justice System*, page 362–373. Longman, 1994.

[Edm17] David Edmonds. Why do people swear? `http://www.bbc.com/news/magazine-39082467`, February 2017. (Retrieved on 2017-08-31).

[Eis05] Elizabeth L. Eisenstein. *The Printing Revolution in Early Modern Europe*. Cambridge University Press, 2005.

[FKS03] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. In *Proceedings of the fourteenth annual ACM-SIAM Symposium on Discrete algorithms*, SODA '03, page 28–36, 2003.

[FLKS17] Gilad Feldman, Huiwen Lian, Michal Kosinski, and David Stillwell. Frankly, We Do Give a Damn. *Social Psychological and Personality Science*, January 2017.

[FPM$^+$09] Clayton Fink, Christine D. Piatko, James Mayfield, Tim Finin, and Justin Martineau. Geolocating Blogs from Their Textual Content. In *AAAI Spring Symposium: Social Semantic Web: Where Web 2.0 Meets Web 3.0*, page 25–26. AAAI, 2009.

[FPR06] Giorgio Fumera, Ignazio Pillai, and Fabio Roli. Spam filtering based on the analysis of text information embedded into images. *Journal of Machine Learning Research*, 7:2006, 2006.

[Gar10] Sean Garrett. Big Goals, Big Game, Big Records. `http://blog.twitter.com/2010/06/big-goals-big-game-big-records.html`, June 2010.

[GCCG11] Roberto Gonzalez, Ruben Cuevas, Angel Cuevas, and Carmen Guerrero. Where are my followers? Understanding the Locality Effect in Twitter. *ArXiv e-prints*, May 2011.

[Ges14] Colleen Geske. *Stuff Dutch People Like: A Humorous Cultural Guide to the Netherlands*. Stuffdutchpeoplelike.com, 2014. (Retrieved on 2017-12-21).

[GL10] Thomas Gottron and Nedim Lipka. A comparison of language identification approaches on short, query-style texts. In *Proceedings of the 32nd European conference on Advances in Information Retrieval (ECIR'2010)*, page 611–614, 2010.

[GLN08] Lena Grothe, Ernesto William De Luca, and Andreas Nürnberger. A Comparative Study on Language Identification Methods. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, 2008.

[Gra10] Tim Grant. Txt 4n6: Idiolect free authorship analysis. In Malcolm Coulthard and Alison Johnson, editors, *Routledge Handbook of Forensic Linguistics*. Routledge, 2010.

[Gri03] Matthew Grimm. When the Sh*t Hits the Fan. *American Demographics*, 25(10), December 2003. Via Tracy V. Wilson *Profanity: Swearing, Cursing and Expeletives*. SHARE, 12 (208), November 2010 `http://www.shareeducation.com.ar/pastissues3/208/1.htm` (Retrieved on 2017-08-31).

[GSR09] Sumit Goswami, Sudeshna Sarkar, and Mayur Rustagi. Stylometric Analysis of Bloggers' Age and Gender. In *International AAAI Conference on Weblogs and Social Media*, 2009.

[GTPZ10] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. @spam: the underground on 140 characters or less. In *Pro-*

*ceedings of the 17th ACM conference on Computer and communications security*, CCS '10, page 27–37, New York, NY, USA, 2010. ACM.

[HAAD+98] B. Habert, G. Adda, M. Adda-Decker, P. Boula de Marëuil, S. Ferrari, O. Ferret, G. Illouz, and P. Paroubek. Towards tokenization evaluation. In Navidad Gallardo Rosa Castro Antonio Rubio and Antonio Tejada, editors, *Proceedings of First International Conference on Language Resources and Evaluation (LREC)*, volume I, page 427–431, Grenada, Spain, 1998.

[Ham50] R. W. Hamming. Error Detecting and Error Correcting Codes. *The Bell System Technical Journal*, 26(2):147–160, 1950.

[HBB+06] Baden Hughes, Timothy Baldwin, Steven Bird, Jeremy Nicholson, and Andrew Mackinlay. Reconsidering language identification for written language resources. In *Proceedings of the fifth International Conference on Language Resources and Evaluation (LREC'2006)*, page 485–488, 2006.

[HCC11] Lichan Hong, Gregorio Convertino, and Ed Chi. Language Matters In Twitter: A Large Scale Study. In *Proceedings of the fifth International AAAI Conference on Weblogs and Social Media (ICWSM'2011)*, page 518–521, 2011.

[Her17] Josh Paul Herron. *Moving Composition: Writing in a Mobile World*. PhD thesis, Clemson University, 2017.

[HF07] Graeme Hirst and Ol'ga Feiguina. Bigrams of Syntactic Labels for Authorship Discrimination of Short Texts. *Literary and Linguistic Computing*, 22(4):405–417, 2007.

[Hir85] Robert Hirsch. Taxonomies of Swearing. In *Perspectives on Swearing*, number 2 in Swearing Report, page 37–59. University of Gothenburg, Dept. of Linguistics, 1985.

[Hub10] HubSpot. State of the Twittersphere. Technical report, HubSpot, January 2010.

[Hum10] Lee Humphreys. Historicizing Microblogging. *CHI 2010*, page 1–4, 2010.

[Ins12] IP Instituto Nacional de Estatística, editor. *Estatísticas Demográficas 2010*. Instituto Nacional de Estatística, Av. António José de Almeida 1000-043 Lisboa Portugal, 2012.

[Jay92] Timothy Jay. *Cursing in America: A Psycholinguistic Study of Dirty Language in the Courts, in the Movies, in the Schoolyards* *and on the Streets*. John Benjamins Publishing Company, January 1992.

[Jay09] Timothy Jay. The utility and ubiquity of taboo words. *Perspectives on Psychological Science*, 4(2):153–161, 2009.

[JJ07] Timothy Jay and Kristin Janschewitz. Filling the emotional gap in linguistic theory: Commentary on Pot's expressive dimension. *Theoretical Linguistics*, 33(2):215–221, 2007.

[JKMdR08] Valentin Jijkoun, Mahboob Alam Khalid, Maarten Marx, and Maarten de Rijke. Named entity normalization in user generated content. In *Proceedings of the second workshop on Analytics for noisy unstructured text data*, AND '08, page 23–30, New York, NY, USA, 2008. ACM.

[JKR$^{+}$99] Varghese Jacob, Ramayya Krishnan, Young U. Ryu, R. Chandrasekaran, and Sungchul Hong. Filtering objectionable internet content. In *Proceedings of the 20th international conference on Information Systems*, ICIS '99, page 274–278, Atlanta, GA, USA, 1999. Association for Information Systems.

[JL08] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, page 219–230, New York, NY, USA, 2008. ACM.

[Joa98] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398 in Lecture Notes in Computer Science, page 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[JZSC09] Bernard J. Jansen, Mimi Zhang, Kate Sobel, and Abdur Chowdury. Twitter power: Tweets as electronic word of mouth. *Journal of the American Society for Information Science and Technology*, 60(11):2169–2188, 2009.

[KF67] Henry Kučera and W. Nelson Francis. *Computational analysis of present-day American English*. Brown University Press, Providence, RI, 1967.

[KM01] Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. In *Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, NAACL '01, page 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[KNF$^+$09] Govind Kothari, Sumit Negi, Tanveer A. Faruquie, Venkatesan T. Chakaravarthy, and L. Venkata Subramaniam. SMS based interface for FAQ retrieval. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL-IJCNLP '09, page 852–860, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[KSA09] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. Computational Methods in Authorship Attribution. *Journal of the American Society for Information Science and Technology*, Volume 60(Issue 1):9–26, January 2009.

[Kuk92] Karen Kukich. Techniques for automatically correcting words in text. *ACM Computer Survey*, 24:377–439, December 1992.

[LBS$^+$13] Gustavo Laboreiro, Matko Bošnjak, Luís Sarmento, Eduarda Mendes Rodrigues, and Eugénio Oliveira. Determining language variant in microblog messages. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing 2013*, page 902–907. Association for Computing Machinery, 2013.

[Lev66] V I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, February 1966.

[LF09] Amanda Lenhart and Susannah Fox. Twitter and status updating, Fall 2009. Technical report, Pew Research Center, October 2009.

[Lju10] Magnus Ljung. *Swearing: A Cross-Cultural Linguistic Study*. Palgrave Macmillan, 2011 edition, November 2010.

[LMB07] Nikola Ljubesic, Nives Mikelic, and Damir Boras. Language Indentification: How to Distinguish Similar Languages? In *Proceedings of 29th International Conference on Information Technology Interfaces (ITI'2007)*, page 541–546, 2007.

[LO14a] Gustavo Laboreiro and Eugénio Oliveira. Avaliaçao de métodos de desofuscação de palavrões. *Linguamática*, 6(2):25–43, December 2014.

[LO14b] Gustavo Laboreiro and Eugénio Oliveira. What we can learn from looking at profanity. In Jorge Baptista, Nuno Mamede, Sara Candeias, Ivandré Paraboni, Thiago A. S. Pardo, and

Maria Graças das Volpe Nunes, editors, *Computational Processing of the Portuguese Language*, volume 8775 of *Lecture Notes in Computer Science*, page 108–113. Springer International Publishing, September 2014. `http://labs.sapo.pt/2014/05/obfuscation-dataset/`.

[Lov17] Robbie Love. Bad language revisited: swearing in the Spoken BNC2014. In *Corpus Linguistics 2017 Conference*. University of Birminham, July 2017.

[LSO11] Gustavo Laboreiro, Luís Sarmento, and Eugénio Oliveira. Identifying Automatic Posting Systems in Microblogs. In Luis Antunes and H. Pinto, editors, *Progress in Artificial Intelligence*, volume 7026 of *Lecture Notes in Computer Science*, page 634–648. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-24769-9_46.

[LSTO10] Gustavo Laboreiro, Luís Sarmento, Jorge Teixeira, and Eugénio Oliveira. Tokenizing micro-blogging messages using a text classification approach. In *Proceedings of the fourth workshop on analytics for noisy unstructured text data*, AND '10, page 81–88, New York, NY, USA, October 2010. ACM. `http://labs.sapo.pt/2011/11/sylvester-ugc-tokenizer/`.

[LWD10] Robert Layton, Paul Watters, and Richard Dazeley. Authorship Attribution for Twitter in 140 Characters or Less. In *Proceedings of the 2010 Second Cybercrime and Trustworthy Computing Workshop*, CTC '10, page 1–8, Washington, DC, USA, 2010. IEEE Computer Society.

[Mad09] Shivendu Madhava. Top 3 Microblogging Plateforms to Promote Your Content. `https://web.archive.org/web/20100729025620/http://www.technozeast.com/top-3-microblogging-plateforms-to-promote-your\-content.html`, May 2009. (Retrieved on 2011-01-23).

[MAK08] Altaf Mahmud, Kazi Zubair Ahmed, and Mumit Khan. Detecting flames and insults in text. Technical report, BRAC University, 2008.

[Mak09] Kevin Makice. *Twitter API: Up and Running*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, first edition, March 2009.

[Mar14] Marco Polo Travel Publishing. *Italian Marco Polo Phrasebook*. Marco Polo Phrasebook. Marco Polo Travel Publishing, Ltd., July 2014.

[MBR12] Diana Maynard, Kalina Bontcheva, and Dominic Rout. Challenges in developing opinion mining tools for social media. In *Proceedings of @NLP can u tag #usergeneratedcontent?!*, Workshop at LREC 2012, May 2012.

[McE06] Tony McEnery. *Swearing in English: Bad Language, Purity and Power from 1586 to the Present*. Routledge Andvances in Corpus Linguistics. Routledge, 2006.

[MK09] Michael Mathioudakis and Nick Koudas. Efficient identification of starters and followers in social media. In *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*, page 708–719, New York, NY, USA, 2009. ACM.

[MK10] Michael Mathioudakis and Nick Koudas. TwitterMonitor: trend detection over the twitter stream. In *Proceedings of the 2010 international conference on Management of data*, SIGMOD '10, page 1155–1158, New York, NY, USA, 2010. ACM.

[MP03] Matthias R. Mehl and James W. Pennebaker. The sounds of social life: A psychometric analysis of students' daily social environments and natural conversations. *Journal of Personality and Social Psychology*, 84(4):857–870, 2003.

[MS99] Christopher D. Manning and Heinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

[MS05] Bruno Martins and Mário J. Silva. Language identification in web pages. In *Proceedings of the 2005 ACM symposium on Applied computing (SAC'05)*, page 764–768, 2005.

[MX03] AM McEnery and Zhonghua Xiao. Fuck revisited. In *Proceedings of the corpus linguistics 2003 conference*, page 504–512. Centre for Computer Corpus Research on Language Technical Papers, 2003.

[MX04] Anthony McEnery and Zhonghua Xiao. Swearing in modern British English: the case of fuck in the BNC. *Language and Literature*, 13(3):235–268, August 2004.

[Nav01] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[NW70] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970. Largest Common Substring.

[NY00] Grace Ngai and David Yarowsky. Rule Writing or Annotation: Cost-efficient Resource Usage For Base Noun Chunking. In *In Proceedings of the 38th Meeting of ACL*, page 117–125, 2000.

[OBRS10] Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, 2010.

[Pia14] Steven T Piantadosi. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic bulletin & review*, 21(5):1112–1130, 2014.

[PJO07] D. Pavelec, E. Justino, and L. S. Oliveira. Author Identification using Stylometric Features. *Inteligencia Artificial, Revista Iberoamericana de IA*, 11(36):59–66, 2007.

[PLZC09] Thomas Park, Jiexun Li, Haozhen Zhao, and Michael Chau. Analyzing Writing Styles of Bloggers with Different Opinions. In *Proceedings of the 19th Annual Workshop on Information Technologies and Systems (WITS'09)*, Phoenix, Arizona, USA, December 14-15, 2009 2009.

[PPSC$^+$12] Daniel Preotiuc-Pietro, Sina Samangooei, Trevor Cohn, Nicholas Gibbins, and Mahesan Niranjan. Trendminer: An Architecture for Real Time Analysis of Social Media Text. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media, Workshop on Real-Time Analysis and Mining of Social Streams (ICWSM'12)*, 06 2012.

[PT 94] PT Staff. Oh, #@*...!! *Psychology Today*, 27(3), May 1994. (Retrieved on 2017-08-31).

[RCD10] Alan Ritter, Colin Cherry, and Bill Dolan. Unsupervised Modeling of Twitter Conversations. In *Proc. HLT-NAACL 2010*, page 172–180, Jun 2010.

[Rea05] Jonathon Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *ACL '05: Proceedings of the ACL Student Research Workshop*, page 43–48, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[RIUM10] Amir Hossein Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. Offensive language detection using multi-level classification. In Atefeh Farzindar and Vlado Keselj, editors, *Advances in Artificial Intelligence*, volume 6085 of *Lecture Notes in Computer Science*, page 16–27. Canadian Conference on Artificial Intelligence, Springer, 2010.

[RKM10] Sindhu Raghavan, Adriana Kovashka, and Raymond Mooney. Authorship attribution using probabilistic context-free grammars. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, page 38–42, Stroudsburg, PA, USA, 2010. ACL, Association for Computational Linguistics.

[RYSG10] Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. Classifying latent user attributes in twitter. In *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, SMUC '10, page 37–44. ACM, 2010.

[SAC12a] Sara Owsley Sood, Judd Antin, and Elizabeth F. Churchill. Profanity use in online communities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, page 1481–1490, New York, NY, USA, 2012. ACM.

[SAC12b] Sara Owsley Sood, Judd Antin, and Elizabeth F. Churchill. Using crowdsourcing to improve profanity detection. In *AAAI Spring Symposium: Wisdom of the Crowd*, volume SS-12-06 of *AAAI Technical Report*. AAAI, 2012.

[SCA11] Sara Owsley Sood, Elizabeth F. Churchill, and Judd Antin. Automatic identification of personal insults on social news sites. *Journal of the American Society for Information Science and Technology*, October 2011.

[Sma09] Roland Smart. The Value of UGC. `http://www.rolandsmart.com/2009/02/the-value-of-ugc/`, February 2009. (Retrieved on 2010-11-23).

[SOM10] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, WWW '10, page 851–860, New York, NY, USA, 2010. ACM.

[Spe97] Ellen Spertus. Smokey: Automatic recognition of hostile messages. In *Proceedings of Innovative Applications of Artificial Intelligence (IAAI)*, page 1058–1065, 1997.

[SRFN09] L. Venkata Subramaniam, Shourya Roy, Tanveer A. Faruquie, and Sumit Negi. A survey of types of text noise and techniques to handle noisy text. In *Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, AND '09, page 115–122, New York, NY, USA, 2009. ACM.

[SSG06] Kaveri Subrahmanyam, David Smahel, and Patricia Greenfield. Connecting Developmental Constructions to the Internet: Identity Presentation and Sexual Exploration in Online Teen Chat Rooms. *Developmental Psychology*, 42(3):395–406, 2006.

[SSLS$^+$11] Rui Sousa-Silva, Gustavo Laboreiro, Luís Sarmento, Tim Grant, Eugénio Oliveira, and Belinda Maia. 'twazn me!!! ;(' Automatic Authorship Analysis of Micro-Blogging Messages. In Rafael Muñoz, Andrés Montoyo, and Elisabeth Métais, editors, *Procedings of the 16th International Conference on Applications of Natural Language to Information Systems, NLDB 2011*, number LNCS 6716 in Lecture Notes in Computer Science, page 161–168. Springer, June 2011.

[SSSG$^+$10] Rui Sousa-Silva, Luís Sarmento, Tim Grant, Eugénio C. Oliveira, and Belinda Maia. Comparing Sentence-Level Features for Authorship Analysis in Portuguese. In *PROPOR*, page 51–54, 2010.

[The08] Mike Thelwall. Fk yea I swear: cursing and gender in MySpace. *Corpora*, 3(1):83–107, 2008.

[TLC11] Yi-Jie Tang, Chang-Ye Li, and Hsin-Hsi Chen. A Comparison between Microblog Corpus and Balanced Corpus from Linguistic and Sentimental Perspectives. In *Workshop on Analyzing Microtext (AAAI'2011)*, 2011.

[TOS16] The 100 most vulgar slang words. `http://onlineslangdictionary.com/lists/most-vulgar-words/`, 2016. (Retrieved on 2016-12-28).

[TSR$^+$07] Hironori Takeuchi, L. Venkata Subramaniam, Shourya Roy, Diwakar Punjani, and Tetsuya Nasukawa. Sentence boundary detection in conversational speech transcripts using noisily labeled examples. *International Journal on Document Analysis and Recognition*, 10(3):147–155, December 2007.

[TSSW10] Andranik Tumasjan, Timm Sprenger, Philipp Sandner, and Isabell Welpe. Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. In *Proceedings of* *the Fourth International AAAI Conference on Weblogs and Social Media*, 2010.

[TWH07] Katrin Tomanek, Joachim Wermter, and Udo Hahn. Sentence and token splitting based on conditional random fields. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, page 49–57, Melbourne, Australia, 2007.

[TWW09] Buzhou Tang, Xuan Wang, and Xiaohong Wang. Chinese Word Segmentation Based on Large Margin Methods. *International Journal of Asian Language Processing*, 19(2):55–68, 2009.

[Ukk92] Esko Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science*, 92(1):191–211, 1992.

[Uni15] Inc. Unicode. Unicode Character Database, v8.0. `http://www.unicode.org/Public/8.0.0/ucd/PropList.txt`, May 2015. (Retrieved on 2018-01-10).

[Vot] Martin Votruba. The word for "bear". `https://www.pitt.edu/~votruba/qsonhist/bearetymologyslovakenglishwelsh.html`. (Retrieved on 2018-01-04).

[VVV10] Tommi Vatanen, Jaakko J. Väyrynen, and Sami Virpioja. Language Identification of Short Text Segments with N-gram Models. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA).

[Wal80] Mort Walker. *The Lexicon of Comicana*. iUniverse, 1980. Via Wikipedia `https://en.wikipedia.org/wiki/The_Lexicon_of_Comicana` (Retrieved on 2017-01-12).

[Wan10] Alex Wang. Detecting Spam Bots in Online Social Networking Sites: A Machine Learning Approach. In Sara Foresti and Sushil Jajodia, editors, *Data and Applications Security and Privacy XXIV*, volume 6166 of *Lecture Notes in Computer Science*, page 335–342. Springer Berlin / Heidelberg, 2010.

[WCTS14] Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P. Sheth. Cursing in English on Twitter. In *Proceed-*

*ings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, February 2014. Updated version at `http://wiki.knoesis.org/index.php/Cursing_in_English_on_Twitter` (Retrieved on 2016-12-05).

[Wil10] Evan Williams. The Evolving Ecosystem. `http://blog.twitter.com/2010/09/evolving-ecosystem.html`, September 2010.

[XFW$^+$12] Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, page 1980–1984. ACM, 2012.

[XZ10] Zhi Xu and Sencun Zhu. Filtering offensive language in online communities using grammatical relations. In *Proceedings of the Seventh Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, 2010.

[Zip35] George Kingsley Zipf. *The Psychobiology of Language*. Houghton-Mifflin, New York, NY, USA, 1935.

[ZP11] Chao Michael Zhang and Vern Paxson. Detecting and Analyzing Automated Activity on Twitter. In *Proceedings of the 12th Passive and Active Measurement Conference (PAM 2011)*, 2011.