Finding the online cry for help

Automatic text classification for suicide prevention

Bart Desmet

UNIVERSITEIT GENT

Proefschrift voorgelegd tot het behalen van de graad van Doctor in de Taalkunde
Promotoren: Prof. dr. Véronique Hoste - Prof. dr. Johan De Caluwe

Faculteit Letteren en Wijsbegeerte

Finding the online cry for help

Automatic text classification for suicide prevention

Automatische tekstclassificatie voor suïcidepreventie op sociale media

Proefschrift voorgelegd tot het behalen van de graad van
Doctor in de Taalkunde aan de Universiteit Gent te verdedigen door

Bart Desmet

Dit onderzoek werd gefinancierd door het
Onderzoeksfonds van de Hogeschool Gent (HOF).

Gent, 2014

Promotoren:
Prof. dr. Véronique Hoste
Prof. dr. Johan De Caluwe

Cover illustration by Gerard Herman.

Abstract

Successful prevention of suicide, a serious public health concern worldwide, hinges on the adequate detection of suicide risk. While online platforms are increasingly used for expressing suicidal thoughts, manually monitoring for such signals of distress is practically infeasible, given the information overload suicide prevention workers are confronted with. In this thesis, the automatic detection of suicide-related messages is studied. It presents the first classification-based approach to online suicidality detection, and focuses on Dutch user-generated content.

In order to evaluate the viability of such a machine learning approach, we developed a gold standard corpus, consisting of message board and blog posts. These were manually labeled according to a newly developed annotation scheme, grounded in suicide prevention practice. The scheme provides for the annotation of a post's relevance to suicide, and the subject and severity of a suicide threat, if any. This allowed us to derive two tasks: the detection of suicide-related posts, and of severe, high-risk content. In a series of experiments, we sought to determine how well these tasks can be carried out automatically, and which information sources and techniques contribute to classification performance.

The experimental results show that both types of messages can be detected with high precision. Therefore, the amount of noise generated by the system is minimal, even on very large datasets, making it usable in a real-world prevention setting. Recall is high for the relevance task, but at around 60%, it is considerably lower for severity. This is mainly attributable to implicit references to

suicide, which often go undetected.

We found a variety of information sources to be informative for both tasks, including token and character ngram bags-of-words, features based on LSA topic models, polarity lexicons and named entity recognition, and suicide-related terms extracted from a background corpus.

To improve classification performance, the models were optimized using feature selection, hyperparameter, or a combination of both. A distributed genetic algorithm approach proved successful in finding good solutions for this complex search problem, and resulted in more robust models. Experiments with cascaded classification of the severity task did not reveal performance benefits over direct classification (in terms of F_1-score), but its structure allows the use of slower, memory-based learning algorithms that considerably improved recall.

At the end of this thesis, we address a problem typical of user-generated content: noise in the form of misspellings, phonetic transcriptions and other deviations from the linguistic norm. We developed an automatic text normalization system, using a cascaded statistical machine translation approach, and applied it to normalize the data for the suicidality detection tasks. Subsequent experiments revealed that, compared to the original data, normalized data resulted in fewer and more informative features, and improved classification performance. This extrinsic evaluation demonstrates the utility of automatic normalization for suicidality detection, and more generally, text classification on user-generated content.

Samenvatting

Zelfdoding is wereldwijd een belangrijke doodsoorzaak, en één van de sleutels tot doeltreffende preventie ligt bij het tijdig herkennen van signalen van slachtoffers. Die maken steeds vaker gebruik van sociale netwerksites om suïcidale gedachten te uiten. Het is voor hulpverleners praktisch onhaalbaar om zorgwekkende berichten manueel op te sporen, omdat de hoeveelheid data op dergelijke platformen te groot is. In dit proefschrift beschrijven we de eerste automatische aanpak voor de detectie van suïcidale berichten door middel van tekstclassificatie. Hierbij ligt de focus op Nederlandstalige sociale media.

Om de haalbaarheid van een dergelijke 'machine learning'-aanpak te kunnen testen, werd een corpus verzameld bestaande uit forummateriaal en blogposts, dat kan dienen als referentie (gouden standaard). Er werd een annotatieschema ontwikkeld om berichten te labelen, vertrekkende vanuit de preventiepraktijk. Dat schema laat onder andere toe om voor elk bericht te bepalen of het met zelfdoding te maken heeft (relevantie), en of er sprake is van een ernstige suïcidale dreiging (intensiteit). Op die manier werden twee detectietaken gedefinieerd. In een reeks experimenten gingen we na in hoeverre deze taken automatisch uitgevoerd kunnen worden, en welke informatiebronnen en technieken de performantie helpen verbeteren.

De experimentele resultaten tonen aan dat beide soorten berichten met een hoge precisie kunnen worden gedetecteerd. Kortom, de hoeveelheid fouten in de suggesties van het systeem (ruis) is verwaarloosbaar, zelfs op heel uitgebreide datasets. Het systeem zou dus bruikbaar zijn voor suïcidepreventie in

een realistische omgeving waarin veel data omgaat. Voor de relevantietaak observeren we ook een hoge recall (een maat voor het aantal berichten dat niet gemist werd). Die is echter aanzienlijk lager voor de intensiteitstaak, met scores rond de 60%. Een kwalitatieve analyse wees uit dat dit vooral veroorzaakt wordt door berichten met impliciete verwijzingen naar zelfdoding, die moeilijk detecteerbaar blijken.

Verschillende informatiebronnen werden geschikt bevonden voor beide taken: bags-of-words van token- en karakter-n-grammen, features gebaseerd op LSA topicmodellen, polariteitlexicons en named entity recognition, en suïcidegerelateerde termen die uit een achtergrondcorpus werden geëxtraheerd.

Om de classificatieperformantie te verbeteren werden de modellen geoptimaliseerd door middel van featureselectie, hyperparameteroptimalisatie of een combinatie van beide. Een aanpak gebaseerd op genetische algoritmes werd gebruikt om geschikte oplossingen te vinden voor deze complexe zoektaak. Dat resulteerde in hogere performantie en robuustere systemen. We experimenteerden ook met getrapte in plaats van directe classificatie voor de intensiteitstaak. Deze aanpak bood geen meerwaarde in termen van precisie en F_1, maar de structuur liet wel toe gebruik te maken van tragere memory-based learning algoritmes die de recall verhoogden.

Op het einde van dit proefschrift behandelen we een probleem dat eigen is aan het taalgebruik op sociale media: ruis in de vorm van spellingsfouten, fonetische transcripties en andere mogelijke afwijkingen van de taalnorm. We ontwikkelden een automatisch normalisatiesysteem, gebaseerd op machinevertaling op het woord- en karakterniveau. Wanneer dit systeem werd toegepast om de data voor de detectietaken te normaliseren, bleek dat genormaliseerde data leidde tot minder en informatievere features, die op hun beurt zorgden voor performantere classificatie. Deze externe evaluatie geeft aan dat automatische normalisatie nuttig kan zijn voor suïcidaliteitsdetectie, en voor tekstclassificatie op sociale media meer in het algemeen.

Acknowledgements

Writing this dissertation has been a healthy challenge, although one could argue about the semantics of healthy. It has also formed the backdrop to an eventful four years, and I am glad for the opportunity to thank everyone who has shared them with me.

I want to start by thanking my supervisor, Véronique Hoste, who has been instrumental to this work, and to my development as a researcher. I have much enjoyed the freedom she gave me to learn new things and explore a variety of problems, which has at times been exhilarating - both intellectually, and whenever close to a deadline. Véronique, thank you for your unwavering trust, optimism and ambition, your generosity and support.

To Johan De Caluwe and Veerle Ongenae, I want to express my sincere gratitude for being in my thesis committee, making the more administrative aspects of the PhD so hassle-free, and for their insightful comments and support. I am also grateful to Walter Daelemans, Iris Hendrickx and John Pestian, who kindly agreed to be in my jury.

I feel very fortunate to have been working on a research topic that is both scientifically interesting and directly relevant to society. I suppose every PhD student questions what he or she is doing at some point, and in such moments of doubt, the clear sense of purpose was a powerful motivator. Also, it made for less awkward conversations about what it is we researchers do exactly (that dreaded moment when the hairdresser is done talking about the weather).

I am greatly indebted to the staff and volunteers at the Centrum ter Preventie van Zelfdoding. A big thank you to Kirsten and Grieke, for the spontaneous collaboration, to Michaël and Jan, for their help in developing the guidelines and providing background, to all annotators and to Imke especially, for her tireless work on managing the annotation effort.

For the more technical aspects of this thesis, I would like to acknowledge Sander's work on extending the BRAT tool, the fruitful collaboration with Jan and David on Gallop, the infrastructure and expert support provided by the High Performance Computing unit and the countless Stack Overflow contributors that have helped me out of many a programming conundrum.

A good many colleagues from Ghent, Antwerp and abroad have livened up lunch conversations, coffee breaks, workshops and conferences. Special thanks should go to the ever-growing list of wonderful people that make LT3 the merry research group that it is: Lieve, with her sound and practical advice, Els, whose infectious laughter and optimism can work wonders, Klaar, for happily (and bravely) sharing an office with me in these past months, Marjan, for her dry wit and our endless discussions about feelings (or rather, sentiments) and Sarah, for all the help (not only on Gallop). Thank you Arda, Cynthia, Joke, Kathelijne, Nils, Mariya and Peter, for being such nice and helpful colleagues, for upholding the birthday cake and other traditions, and even venturing to introduce some new ones (quidditch comes to mind). Orphée and Isabelle, I could not have wished for a better office mate and friend-turned-colleague. Our common history is not limited to the *Departement*: pub nights, barbecues, travels, getting degrees, first paychecks, sandwiches and organized, adjusting to the working life, building homes and writing theses,... It has been an absolute joy to share laughs, successes and setbacks with you, in and out of the office.

Stef, Gerard, Eva, Rocky and Maurits, I feel privileged to have you as friends. With you, I never need to worry about the road not taken. As Robert Frost would have it, we take the one less traveled by, and it makes all the difference. Louise, thank you for the beautiful years, and for your support. Thank you Evelien, Pieter, Mauri and many others from JNM and elsewhere, for your friendship and for providing endless opportunities for fun, food and conversation, worthy causes and adventure.

Finally, I want to thank my family, who have been there all along. Peter and Lien, it's wonderful how we keep moving closer to each other. Markus and Kobe are growing up to be so clever and cheerful, and I look forward to witnessing it from the first row. Sofie and Joren, you have supported me in countless ways, and I hope to do the same. Thank you, mama en papa, for giving us such a carefree childhood and caring about the future, for teaching us not to avoid challenges, and for never losing sight of the important things in life.

Contents

1 Introduction **1**

1.1 Rationale 1

1.2 Background 2

1.2.1 Suicide mortality 2

1.2.2 Suicide pathology and prevention 3

1.2.3 The role of the internet in suicidal behaviour 5

1.2.4 Hurdles to online prevention 7

1.3 Motivation 8

1.4 Research objectives 9

1.5 Thesis outline 10

2 Related research **13**

2.1 Technology-based suicide prevention 13

2.2 Suicide note analysis 16

2.2.1 The language and content of suicide notes 16
2.2.2 Suicide notes and machine learning 18
2.3 Suicidality modelling 19

3 Resources **23**
3.1 Corpora 24
3.1.1 Suicide-related text corpora 25
3.1.2 Reference corpora 26
3.1.3 Problems with the Netlog corpora 27
3.2 Annotation 30
3.2.1 Annotation scheme 30
3.2.2 Annotation implementation 36
3.2.3 Task definition 38
3.2.4 Inter-annotator agreement 39
3.2.5 Annotation statistics 43
3.3 Summary 45

4 Information sources **47**
4.1 Preprocessing 48
4.2 Bag-of-words features 51
4.3 Lexicon-based features 54
4.3.1 Polarity lexicons 54
4.3.2 Term extraction 55
4.4 Topic model features 57
4.5 Other features 61
4.6 Feature counts 62

4.7 Summary 63

5 Machine learning techniques **65**

5.1 Support vector machines 67

5.1.1 Theory 67

5.1.2 Implementation 69

5.2 Memory-based Learning 71

5.2.1 Theory 71

5.2.2 Implementation 74

5.3 Model validation 75

5.4 Direct versus cascaded classification 78

5.5 Summary 80

6 Learner optimization with genetic algorithms **81**

6.1 Hyperparameter optimization 82

6.2 Feature selection 83

6.2.1 Feature filtering 84

6.2.2 Wrapped feature selection 87

6.3 The search problem of optimization 88

6.4 Genetic algorithms 90

6.4.1 Theory 90

6.4.2 Implementation 95

6.4.3 Distributed GA optimization with Gallop 96

6.5 Summary 98

7 Experiments **101**

7.1 Experimental setup 102

7.1.1 Corpus sampling 102
7.1.2 Experimental setup per task 104
7.1.3 k-nearest fitness 106
7.1.4 Baseline classifier 107
7.2 Relevance classification 108
7.2.1 Cross-validation results 108
7.2.2 Results on the held-out testset 112
7.2.3 Selected hyperparameters and features 114
7.3 Direct severity classification 119
7.3.1 Cross-validation results 119
7.3.2 Results on the held-out testset 123
7.3.3 Selected hyperparameters and features 125
7.4 Cascaded severity classification 129
7.4.1 Results with gold standard relevance labels 129
7.4.2 Results with predicted relevance labels 134
7.5 Scaling and error analysis 136
7.5.1 Relevance 137
7.5.2 Severity 140
7.6 Summary 142

8 Normalization **143**
8.1 Related research 144
8.2 Methodology 146
8.2.1 Corpus compilation 146
8.2.2 Annotation 148
8.2.3 System architecture 149

8.3 Experiments and discussion 150
8.3.1 Experimental setup 150
8.3.2 Results on SMS 151
8.3.3 Results on all genres 153
8.3.4 Error analysis 154
8.3.5 Summary 156
8.4 Normalization and suicide detection 157
8.4.1 Normalization of the data 157
8.4.2 Feature extraction 160
8.4.3 Classification tasks 161
8.4.4 Summary 162

9 Conclusion **165**
9.1 Resources 166
9.2 System architecture 166
9.3 Experimental observations 167
9.4 Future work 171

A Publications **175**

B Multiword suicide lexicon **179**

References **190**

CHAPTER 1

Introduction

1.1 Rationale

Suicidal behaviour is a serious public health concern worldwide. The success of suicide prevention hinges on adequate risk assessment and timely support. Online platforms are increasingly used for expressing suicidal thoughts, but those are not necessarily recognized, reported or responded to by platform users or administrators. Expert monitoring by suicide prevention workers, on the other hand, is practically infeasible given the information overload they would be faced with.

In this thesis, we investigate a natural language processing approach for automatically detecting Dutch user-generated text content related to suicide, which may serve to improve the coverage and speed of suicide prevention efforts.

1.2 Background

1.2.1 Suicide mortality

Figures of the World Health Organization (World Health Organization 2011) indicate that globally, 782 000 people died by self-inflicted intentional injuries in 2008, making suicide the sixth leading cause of death for adults aged 20 to 59 years (Mathers et al. 2009). However, suicide mortality is likely underreported, since injury deaths of unknown intent or cause are believed to hide some suicides as well. One million deaths per year is therefore generally accepted as a global estimate (Värnik 2012).

In 2010, suicide was the third leading cause of death in US citizens aged between 1 and 44 years (Miniño and Murphy 2012). In Europe, it is the most prevalent cause of death through injury (at 1.4% mortality), claiming more victims than road traffic accidents and violent crimes. In the Dutch-speaking world, suicide mortality is significantly higher in Flanders than in the Netherlands. Each year, close to one thousand Flemish individuals die by suicide (compared to 1 500 in the 2.5 times larger Dutch population). This corresponds to a mortality rate of 25 per 100 000 in males and 13 per 100 000 in females, figures that are respectively around 50% and 60% lower in the Netherlands (Reynders et al. 2011, Kerkhof 2012). Suicide is the primary cause of death among Flemish teenagers, male adults between 20 and 49 years old, and women aged between 20 and 39.

Suicides have a significant economic impact, and more importantly, they deeply affect the lives of those who lose a relative or friend. In addition to the one million deaths globally, there are an estimated ten to twenty million non-fatal suicide attempts each year. These cause mental and physical suffering for the attempter, and have a disruptive impact on their environment. A study by De Jaegere et al. (2012) on suicide attempts in Flanders found that in 2012, there was an incidence rate of 145 per 100 000 in men and 187 per 100 000 in women. These figures reveal that for every successful suicide, there are almost ten non-fatal attempts, and that attempts are significantly more common in women. A periodical survey on Belgian health (Gisle 2008) corroborates that more women have attempted suicide in their lifetime, and it additionally indicates that suicidal thoughts are more prevalent in women and young people. The results show no correlation between age and the incidence of attempted suicide, but suicidal thoughts are found to be more common in young people, affecting as much as 10% of the male and 15% of the female population between 15 and 24 years old.

1.2.2 Suicide pathology and prevention

As discussed above, suicidality does not manifest itself as an 'all or nothing' phenomenon. Beside the actual deaths by suicide, there is a spectrum of suicidal behaviour that includes suicide attempts, suicide ideation (the tendency to consider ending one's life, i.e. have suicidal thoughts, actively search for information, etc.) and deliberate self-harm. In the suicidological literature, these stadia are typically conceptualized as a suicidal process, which can serve as an explanatory model (Hawton and van Heeringen 2000, van Heeringen 2007, Neeleman 2007).

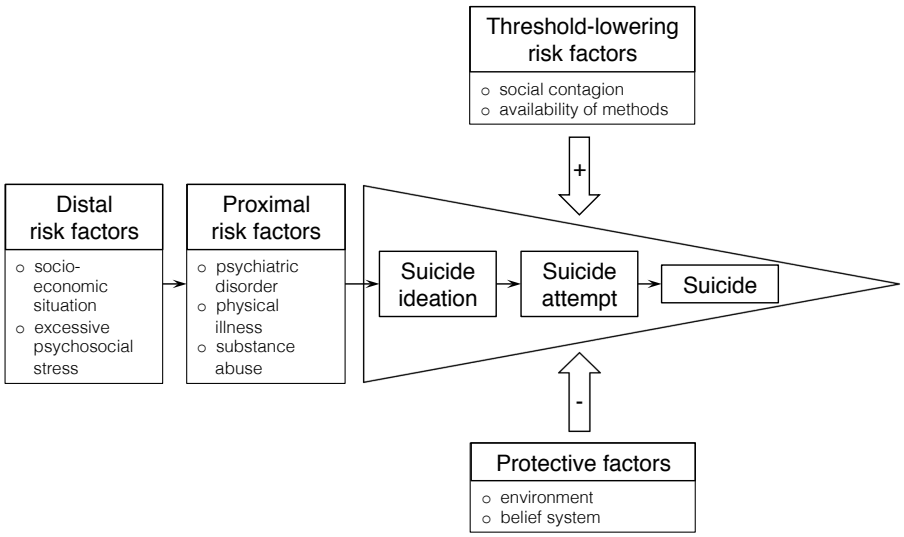

Figure 1.1: Schematic overview of the suicidal process.

Figure 1.1 provides a schematic overview of this model. Suicide is considered to be weakly influenced by so-called distal risk factors, underlying vulnerabilities that do not cause suicidal behaviour, but rather indicate a higher probability for it to occur. Examples of distal risk factors can be found in a person's socio-economic situation (e.g. unemployment, debt) and in psychosocial stress conditions, such as loss of a loved one, social isolation, bullying or abuse.

Secondly, there are proximal risk factors, which directly cause vulnerability for suicidal behaviour. These include biological factors (such as hormonal imbalance or genetic predisposition towards a psychiatric disorder), psychological factors (e.g. impulsivity, low self-esteem, hopelessness or impaired problem-solving ability) and psychiatric conditions like depression, drug addiction or borderline

personality disorder.

In sum, these distal and proximal factors may cause a person to exhibit suicidal behaviour. This starts with suicidal ideation, and may progress to the formulation of a plan, an attempt, or possibly suicide. Whether or not the suicidal process escalates depends in part on the presence of threshold-lowering risk factors, which make escalation more likely, and protective factors, which may reduce the suicide risk. Examples of protective factors include strong relationships (e.g. with children), fear of dying, religious or cultural beliefs or a sense of purpose.

Two important categories of threshold-lowering risk factors are the availability of methods and social contagion. Access to or knowledge about specific methods (e.g. firearms or sedatives) has been shown to increase the probability of those methods being used (Marzuk et al. 1994), and detailed descriptions of a method in the media have been suggested to lead to its increased use in the general population, typically after the death of a celebrity (Phillips and Carstensen 1986, Eggermont et al. 2007). Similarly, social contagion is the phenomenon where suicide is regarded as a valid solution to a potential victim's problems, when he can identify with the conditions surrounding the suicide of a peer or celebrity. This problem is exacerbated when media reports prominently cover suicide, and simplify or romanticize its motives. The copycat behaviour attributed to social contagion is known as the 'Werther effect', named after the protagonist of Goethe's novel *The Sorrows of Young Werther*, believed to have triggered a wave of suicides in 18th century Europe. It is therefore advised that media coverage of suicides be limited, vague with regard to methods, and that it contains a reference to support for those contemplating suicide.

Suicide is generally considered a preventable death, and regardless of a victim's stage in the suicidal process, there often remains ambivalence between life and death. It is a common adage in prevention discourse that *suicide is a permanent solution to a temporary problem*, although some of the reported reasons for committing suicide are not transitory in nature, such as incurable disease or lifelong mental illness. Suicide prevention can be divided into three categories (Neeleman 2007):

- Primary prevention is aimed at reducing suicide risk in the general population. This includes people who have not shown external signs of suicidal behaviour, but who may have suicidal thoughts. Prevention strategies include the reduction of risk factors (e.g. by improving the socio-economic environment, removing barriers to mental health access or regulating the availability of methods), and education of the general public and health professionals to better recognize signs of suicidal behaviour, and how to

seek help.

- Secondary prevention is concerned with patients who are known to have suicidal tendencies or who are at an increased risk of presenting it (e.g. previous attempters or people suffering from depression). This kind of prevention is typically focused on adequate risk assessment, medication and therapy. It also includes acute crisis support such as suicide hotlines, which can add perspective to a victim's situation and refer to specialized help.
- Tertiary prevention deals with the minimization of harmful effects of suicidal behaviour on others, such as professional caregivers, persons involved in the decease (e.g. train drivers, witnesses) and surviving relatives.

1.2.3 The role of the internet in suicidal behaviour

The adoption of the internet and the subsequent rise of Web 2.0 has had far-reaching implications for human communication. Originally, the web consisted of static web sites, which revolutionized the access to information, as it was no longer bound to specific times or locations. The advent of sites that allowed user-generated content, i.e. Web 2.0, blurred the boundary between content providers and consumers, and opened up the possibility to interact and form communities online. Forums or message boards, blogs and social networking platforms such as Facebook or Twitter have become an important means of interpersonal and community-level communication. Inevitably, these developments have also had an impact on the available information on suicide, and how people communicate about suicidal behaviour.

Web sites that provide information about treatment resources, self-help and resources for helping others are an example of how the placeless nature of the internet provides opportunities for suicide prevention. Persons in crisis can access information at all times, and are not limited to seeking help during conventional business hours. Additionally, some users prefer the privacy and anonymity the internet offers (Luxton et al. 2011). However, the same is true for pro-suicide sites, which may offer motivations for suicide or provide information on techniques.

Media coverage of suicides has also shifted. Traditional media, such as print or television, can be requested to follow the recommended reporting guidelines, although getting buy-in from industry can be difficult. With social media, however, anyone can go viral, and it is virtually impossible to educate influencers. Figure 1.2 shows a recent example of a tweet by the Academy of Motion Picture Arts and Sciences, following the suicide of actor Robin Williams. It was

Figure 1.2: Popular Twitter post following the death of actor Robin Williams, who voiced the genie character in Disney's *Alladin*.

shared by hundreds of thousands of Twitter users, and viewed by millions. Suicide prevention experts argued that it presents suicide in too celebratory a light, with the written implication that suicide is somehow a liberating option (Dewey 2014).

Social media have become an outlet for people contemplating suicide to share their thoughts and feelings as well. Particular advantages of online communication are that it can offer anonymity (e.g. on message boards) and a sense of control, since one can choose whether or not to engage with others. Peter et al. (2005) found evidence of this disinhibition, noting that introverted adolescents are strongly motivated to communicate online to compensate for lacking social skills, which led to more online self-disclosure.

Like offline communities, online communities can be harmful. With regard to suicide, two phenomena are of particular interest: cyberbullying and suicide clusters. Unlike traditional bullying, cyberbullying can continue outside of a school or work environment, and new forms have developed that are specific to online communication (Cappadocia et al. 2013). Teenagers have been found to be especially vulnerable to the effects of bullying on social networks (Allison and Schultz 2001), and its effects have been linked to suicide (Hinduja and

Patchin 2010). Online social networking may also play a role in suicide pacts and suicide clusters, where several suicides occur more closely together than would be typically expected, and are believed to be connected through social contagion (Robertson et al. 2012).

1.2.4 Hurdles to online prevention

Online platforms are being used to communicate about suicide, and to express suicidal thoughts. Such feelings can be voiced explicitly, such as in Example 1, or more implicitly (Example 2).

(1) NL: Ik wil gewoon niet meer in leven zijn.
EN: I just don't want to be alive anymore.

(2) NL: Ik hoef me nooit meer te schamen.
Ik hoef nooit meer verdriet te hebben.
Ik hoef nooit iets meer.
EN: I never have to be ashamed again.
I never have to feel sadness again.
I never have to do anything anymore.

Suicidal expressions can be recognized by peers or website administrators, who can respond to the victim, report the content to the online service provider or to an official suicide prevention institution. However, online cries for help may not always be recognized or dealt with in an appropriate or timely fashion. It may therefore be desirable to have suicide prevention workers monitor publicly available user-generated content, or private content if this is not in conflict with users' preferences, safety and privacy concerns.

There are a number of obstacles for prevention stakeholders to monitor user-generated content. Manual monitoring is practically infeasible, given the massive volume of content that is continually produced, a problem referred to as *information overload*. Using keyword searches to locate relevant content would reduce the volume, but still presents a number of problems:

- Specific search queries (e.g. *suicide* or *kill myself*) may only cover a limited range of explicit suicidal expressions. The above posts, for example, would not be retrieved with straightforward search terms.

- Multiple or broader search terms inevitably return false positives (i.e. irrelevant search results), which would increase the burden for prevention workers who monitor the results.

- User-generated content tends to deviate from the linguistic norm. Typical problems include misspellings, the use of abbreviations, phonetic text and colloquial or ungrammatical language use. This may hinder keyword retrieval considerably.

There is a need for more intelligent techniques to automatically filter content based on its relevance to suicide.

1.3 Motivation

The challenge of handling big data is not a new one. There is a large body of work on data mining, research concerned with the discovery of patterns or knowledge in data sources, such as databases, web usage data or texts. There is a variety of natural language processing tasks that are specifically concerned with structuring and mining information from text. Examples include information retrieval, the task of obtaining text resources relevant to an information need (e.g. search engines), text categorization, the task of assigning predefined categories to documents (e.g. spam filtering) and sentiment mining, the field of study concerned with detecting and analysing emotions and opinions, rather than facts, in text (Joachims 1998, Liu 2010, 2011).

Advances in these fields have important economic implications. A model that can successfully categorize the topic of a social media message or model the opinions and preferences of a user, for example, can be used to display advertisements that are more likely to be clicked and generate revenue. This is the business model of large web companies like Google and Facebook. However, there is also significant potential for the technologies to be used for social ends.

In this thesis, we aim to model text about suicide using current natural language processing techniques. More specifically, we intend to approach the problem of detecting user-generated content related to suicide as a text classification task. We investigate the feasibility of making such an application for Dutch, and evaluate its performance.

1.4 Research objectives

In accordance with the research aim described above, the main research question of this study can be formulated as follows:

> **Can relevant and severe messages about suicide be automatically detected in Dutch user-generated content, and if so, which information sources and techniques contribute to classification performance?**

In order to answer this question, we define the following research objectives:

1. **Collect and annotate datasets for the automatic detection of suicidal text.**

   An important challenge in suicide research is the availability of suitable data. For the task at hand, no data is publicly available. There is a need to collect text corpora of Dutch social media messages, containing both suicidal and non-suicidal content. Furthermore, a methodology needs to be developed to consistently annotate these messages. We propose, evaluate and apply an annotation scheme that is grounded in suicide prevention practice.
   From the annotation scheme, two classification tasks are derived: (1) the detection of relevant posts, i.e. posts that are about suicide, either in general or specifically about the author or a third person, and (2) the detection of posts presenting a severe suicide threat, which should receive priority attention from suicide prevention workers.

2. **Develop a model for the detection of suicide-related messages.**

   We investigate the feasibility of automatic relevance detection. A range of information sources (or features) is developed that is potentially useful in modelling suicidal content. We evaluate model performance and investigate which information sources are appropriate for the task.

3. **Develop a model for the detection of severe suicidal messages.**

   Likewise, we test the feasibility of automatic severity detection and analyse the appropriateness of various information sources.

4. **Investigate optimization techniques to maximize model performance.**

   Given the available information sources, we seek to optimally exploit their discriminative power and explore the limits of what model performance

can be achieved with them. To that end, we use two machine learning algorithms, perform model optimization and experiment with cascaded model construction.

5. **Evaluate classification performance on realistic datasets with high class skew.**

   The occurrence of suicide-related content in social media messages is relatively rare. As a result, the proportion of positive (i.e. pertinent) messages is low, and a randomly sampled corpus would exhibit high skew towards the negative class. For the modelling experiments, more balanced datasets are used. It is worthwhile, however, to also evaluate the performance and errors on realistic datasets.

6. **Evaluate the impact of automatic text normalization on classification performance.**

   Given the noisy nature of the user-generated content in which we aim to detect suicidal material, we hypothesize that model performance can be improved by bringing noise closer to the linguistic norm. We develop an automatic system for text normalization, and gauge its impact on classification accuracy.

1.5 Thesis outline

This thesis is divided into nine chapters, which are structured as follows. Chapter 2 discusses existing work on technology-based suicide prevention programmes, and more specifically, the use of natural language processing in the domain of suicide research and prevention.

Chapter 3 introduces the corpora of Dutch user-generated content that were used in this study, and how suicide-related material was obtained. It also discusses the annotation scheme for defining suicide-related and severe posts, and how it was applied to the corpora.

Chapter 4 deals with the information sources or *features* that were developed for the classification tasks. It describes how the data was first cleaned and preprocessed, and how various feature groups were then extracted. Information sources include features based on word and lemma sequences (bags-of-words), sentiment lexicons, relevant term lists, topic models, and miscellaneous semantic and orthographic features.

In Chapter 5, we discuss the various model options that have been tested for the task of automatically detecting suicidal content, and how those models were

validated. We introduce support vector machines (SVM) and memory-based learning (MBL), the two machine learning frameworks used in the experiments, and the possibility to do cascaded classification, in which posts are filtered with a sequence of classifiers.

The methodology for model optimization is the topic of Chapter 6. We applied feature selection through filtering and wrapping, separate hyperparameter optimization and joint optimization. We introduce Gallop, an experimental framework for model optimization that combines genetic algorithms with distributed computing.

The experimental setup and all experimental results are presented and discussed in Chapter 7. We establish to what extent suicide-related and severe posts can be detected, and what the impact is of the various models and optimization strategies. The chapter concludes with scaling experiments: the best-performing models are applied to large datasets with high class skew to simulate performance in a realistic environment, and a qualitative error analysis is conducted.

Chapter 8 explores the potential benefits of text normalization for natural language processing tasks on noisy user-generated content. To that end, we describe a state-of-the-art normalization system that was developed for Dutch. The system was applied to our experimental corpus, and we describe the impact on the feature extraction and classification performance for the task of suicide detection.

Chapter 9 presents the conclusions of this thesis and provides perspectives for future work.

CHAPTER 2

Related research

In this chapter, we give an overview of the existing work on technology-based suicide prevention and the language associated with suicidality. In Section 2.1, we discuss technological approaches for suicide prevention. Section 2.2 presents research on suicide notes, arguably the most prototypical (albeit rare) textual expression of the suicide victim. Section 2.3 concludes this chapter with a discussion of computational approaches to suicidality modelling and prediction.

2.1 Technology-based suicide prevention

Traditional approaches to suicide prevention have relied on the use of brochures, billboards, radio and television for increasing awareness (primary prevention), and on telephone and face-to-face counseling for support to suicidal persons (secondary prevention). With the development and increasing use of new technologies like the Internet and mobile devices, new opportunities have arisen for suicide outreach and prevention.

Technology-based suicide prevention (TBSP) programmes can overcome significant barriers to care (Luxton et al. 2011). The role of time and distance is

minimized: technology can reach remote and underserved communities, and it provides instantaneous and round-the-clock access to information and support. Technology also offers the advantage of anonymity, which may render TBSP services more accessible to people who would otherwise avoid conventional services, because of social stigma or privacy concerns.

A variety of TBSP approaches exist, and they can be organized into three categories: passive, active and interactive forms of prevention (De Jaegere et al. 2013).

Passive forms of TBSP require a user only to read, look or listen to content, without active involvement. These programs typically provide information about warning signs of suicide, self-help and treatment resources, statistics, etc. Content can be provided as text or audiovisual media through web pages, blogs, video-sharing websites or podcasts. Subscription-based services offer the advantage that new content is effortlessly delivered, and may be stored for offline availability.

Banners and links to support may be provided on websites, search engines or e-mail services, if they are used to access suicide-related material. This requires some form of text analysis, such as the Google AdWords engine that provides context-aware advertisements.

E-mail outreach is an intervention method that has been proved effective for preventing suicide. It involves periodically sending personalized correspondence to patients, after they have been discharged from psychiatric treatment. Such patients have a heightened risk for repeated suicide attempt, which is significantly reduced when they are followed up with 'caring letters' (Motto and Bostrom 2001).

Active forms of TBSP expect more involvement and effort from the user. Examples include self-assessment tests, chat bots and serious games that provide feedback, refer to support or teach coping mechanisms. Some social networking platforms (Facebook and Twitter, among others) allow users to report suicidal content.

An interesting new development is the use of mobile phone applications (*apps*) for self-help. Mobile phones have the benefit that they are personal, discreet and readily available. Apps can be designed to help users self-assess and monitor suicidal symptoms, allow them to keep a diary or draw up a safety plan, and provide quick access to information, hotlines, relaxation exercises, a user's favourite music, pictures, contacts, etc. On Track Again[1] is an example of such an application, developed for young people who have survived a suicide attempt.

[1] http://www.ontrackagain.be

Interactive forms of TBSP are characterized by interactions between users (and caregivers). Hotline services for persons in crisis are a typical example. Traditionally, these services are accessible by phone, but such immediate personal contact dissuades some victims from contacting them. Online chat offers a means of communication that is more accessible and familiar to certain groups of users, especially younger ones. E-mail correspondence is another, more asynchronous option.

Peer-based interactive TBSP involves platforms that do not require the intervention of prevention experts, such as forums and group chats. An advantage of social networking sites is that they facilitate social connections among peers with similar experiences. They can foster supportive interactions and create a community among those who are coping with similar challenges.

In sum, technology can play an important role in suicide prevention, and its importance continues to grow. However, technological solutions will always need to be complemented and motivated by clinical practice. Furthermore, they should be developed to meet acceptable standards in terms of quality, transparency and security (De Jaegere et al. 2013).

Online risk detection

The text classification approach we propose in this study is intended to detect suicide-related content in social media postings, e-mails or other textual content. Such technology could allow large-scale automated monitoring of online content, to identify users who might be at risk for suicide. It can therefore be considered a passive form of primary prevention, aimed at connecting caregivers to potential victims for secondary prevention.

Although no such applications currently exist, there is prior research on the automatic detection of other kinds of harmful online content that demonstrate the potential of text analysis for online risk mitigation. This includes work on the automatic detection of cyberbullying victims and perpetrators (Yin et al. 2009, Reynolds et al. 2011, Dadvar 2014), offensive language (Razavi et al. 2010, Xiang et al. 2012), flaming, i.e. hostile interactions (Alonzo and Aiken 2004), racism (Munezero et al. 2011), and sexual predation (Pendar 2007, McGhee et al. 2011, Peersman et al. 2012).

In the AMiCA project[2], for example, a platform is being developed for the cross-media detection of online threats in text and visual media, particularly in relation to child safety (cyberbullying, self-harm and sexually transgressive

[2]http://www.amicaproject.be/

behaviour). It investigates the feasibility of detection, threat monitoring and response (e.g. through dashboard applications for moderators or alerter services), trend analysis, and the technical challenges associated with handling large amounts of data. It also addresses the ethical and legal challenges involved in developing and deploying such a system, and the user requirements and desirability of online monitoring (Van Royen et al. 2014).

2.2 Suicide note analysis

Research conducted on the topic of 'suicidal text' has revolved primarily around suicide notes, one of the few textual resources that provide an insight into the suicidal frame of mind. For this reason, the genre has been studied in great detail from psychological and psychiatric perspectives.

2.2.1 The language and content of suicide notes

Early work on suicide notes dates back to the 1950s, when Shneidman and Farberow (1957) recognized that suicidal persons are not inherently insane, a view commonly held at the time, but rather that suicidal behaviour is motivated by many factors, including psychological ones. They studied these factors in psychiatric case histories, psychological test results and suicide notes. To that end, a corpus of 721 suicide notes was obtained from the Los Angeles County Coronor's Office, dated between 1944 and 1953. As control data, they also elicited simulated suicide notes from individuals who were assumed non-suicidal after careful screening. The fake notes were matched to the genuine notes in terms of author age and occupational level, yielding 33 pairs of notes. The resulting corpus of 66 notes served as the raw material for most of the suicide note research in the subsequent decades, discussed below. Methodological concerns have been voiced concerning the preselection of simulated note writers (Black 1993) and the representativeness of the author sample, which was all male, Caucasian, Protestant, American and between 25 and 59 years of age (Shapero 2011).

Originally, research was focused on identifying features to differentiate between genuine and fake notes. Shneidman and Farberow (1957) apply a discourse analytic method, using the *Discomfort-Relief Quotient*, in which text sequences ('thought units') are classified as expressing negative emotions (e.g. guilt, blame or aggression), positive (e.g. love) or neither (neutral). Genuine notes are found to present more intense discomfort, and more neutral expressions than simulated ones.

A more linguistic approach is taken in Osgood and Walker (1959). They compare genuine suicide notes to fake notes and to ordinary letters to friends and family, the latter comparison being of particular interest to our study. The authors hypothesize that suicide note writers function under heightened motivation, and that this is reflected in the structure and content they produce. They find that suicide notes display greater stereotypy, reflected in shorter and simpler words, a less diverse vocabulary, more repetitions and so-called 'allness' terms like *always*, *never*, *everyone* or *completely*. They also contain significantly more evaluative terms (e.g. *unfair*, *drunkard*), time designations, directives (e.g. *Don't feel too bad about this*) and expressions of ambivalence (e.g. *but*, *should*, *possibly*, *seems*). No evidence is found for increased disorganization, measured in terms of orthographic or grammatical errors and average sentence length.

Gottschalk and Gleser (1960) aim to measure a 'wide range of personality variables through the systematic analysis of verbal behaviour' in the corpus. In essence, this analysis focuses on the use of grammatical word categories (e.g. verbs or adverbs) and a predefined set of 'themes' (e.g. emotions, perceptions, material things). They find no significant trends in terms of grammatical categories, but observe that genuine notes have more references to people, places and things, and fewer to cognitive processes. This 'greater specificity' is confirmed in the study by Ogilvie et al. (1966), who perform the first computational analysis of suicide notes, using software to allocate words into semantic categories pertinent to psychology and sociology. In order to 'create a profile of suicidal language', Edelman and Renshaw (1982) use software to tag the notes with grammatical and psychological categories similar to Gottschalk and Gleser. They also include a form of polarity analysis: genuine notes are found to contain more negative references to the concept 'you' and to specific people, places and things, and more positive references to third persons.

Leenaars (1988) analyzed the corpus using one hundred statements describing potential content of suicide notes. Five such topics were highly predictive of genuine suicide notes: distress and grief, pain, illness and rejection, 'underdeveloped personality organization' and contradictions. The study also included more recent notes, and investigates the influence of author age and sex, finding only age-related differences.

All the above studies are corpus-based, but they come from the discipline of psychology. A comprehensive linguistic study is presented in Shapero (2011). It focuses on a corpus of 286 British notes to determine the characteristics of the suicide note genre. The distribution of lexical items and semantic categories is studied within and between texts. This shows that there is specific vocabulary associated with the beginnings and endings of notes, and that compared to other texts, suicide notes are distinctively about affection, references to the future, the authors' knowledge and their relatives. They also exhibit an above-

average frequency of pronouns, proper names, misspellings, negatives, discourse markers, maximum quantity terms and intensifiers. Additionally, the 66-note corpus of Shneidman and Farberow is approached from a forensic linguistics perspective, i.e. with the objective of defining a litmus test for the authenticity of a suicide note, to be used for example in criminal investigations. Shapero concludes that while the probability of a note being genuine or fake can be estimated by the presence or absence of certain features, absolute predictors are unlikely to exist.

2.2.2 Suicide notes and machine learning

In recent years, an increasing number of studies have applied machine learning techniques to model aspects of suicidality. Pestian et al. (2008) were the first to apply supervised machine learning to the task of distinguishing genuine from fake suicide notes (data from Shneidman and Farberow), and expand on their work in Pestian et al. (2010). A range of features was extracted from the data, including statistics about parts-of-speech, sentence length and parse tree depth, two readability scores, frequent words and manually annotated emotional concepts. After filtering on frequency and information gain (cf. page 84), 66 features were selected: 42 words, 18 part-of-speech tags, 4 emotional concepts and both readability scores. A variety of machine learning algorithms was trained and tested using bootstrap resampling, and their performance was compared to that of human raters (mental health professionals and psychiatry trainees). Most algorithms were found to significantly outperform humans, implying that machine learning is relevant for clinical decision support with regard to suicidality.

In Matykiewicz et al. (2009), unsupervised clustering techniques are found to be effective in separating suicide notes from online newsgroup postings, which serve as a proxy for general discourse. Although this may indicate that clustering is helpful in distinguishing suicidal from non-suicidal discourse, we feel that the datasets are too distinct to conclude that the algorithms were modelling suicidality, rather than genre.

Perhaps the most intensively researched topic is that of the detection of emotional concepts in suicide notes. A corpus of 900 genuine suicide notes, annotated with fine-grained emotions, was released in the framework of the 2011 i2b2 NLP Challenge on emotion classification (Pestian et al. 2012), allowing research on which emotions might be indicative of suicidal behavior, and how they can be found automatically. The task was to detect the presence of 15 emotions at the sentence level: abuse, anger, blame, fear, guilt, hopelessness, sorrow, forgiveness, happiness, peacefulness, hopefulness, love, pride, thankful-

ness, instructions, and information. Twenty-four teams participated in the challenge, with many systems performing at levels approaching the inter-annotator agreement, suggesting that human-like performance is within the reach of currently available technologies. Most systems used support vector machines (see page 67), and successful features included word and character ngrams, part-of-speech categories, emotion and polarity lexicons, WordNet lexical domains and hand-crafted patterns (we refer to Chapter 4 for further information on features). Our own submission (Desmet and Hoste 2013a), which ranked tenth in the competition, combined binary support vector machines with ngram, subjectivity and WordNet features, and experimented with spelling correction to reduce lexical variation. This work inspired some of the methodological decisions made in this dissertation, in which we also focus on lexical and semantic features, and consider text normalization as a method to improve classification performance.

2.3 Suicidality modelling

The work on suicide notes aside, machine learning and natural language processing have considerable potential for modelling and detecting suicidality. A number of studies have applied these techniques in other contexts.

In works of art

Stirman and Pennebaker (2001) investigated the oeuvre of suicidal and non-suicidal poets, and tested two models of suicide: a social disengagement model, which hypothesizes that suicidal individuals detach from the source of their pain, withdraw from social relationships and become more self-oriented, and a more traditional model of hopelessness, which suggests that suicide takes place during extended periods of sadness and desperation. The authors used the Linguistic Inquiry and Word Count (Pennebaker et al. 2001, LIWC) text analysis tool to build linguistic profiles of the early, middle and late periods of each poet. LIWC assigns words to various categories (e.g. *cried* → SADNESS, NEGATIVE EMOTION, OVERALL AFFECT and PAST TENSE VERB). Tentative evidence is found for the social disengagement model: the combined late work of suicidal poets is positively correlated to first-person singular self-references and words associated with death, and negatively to first-person plural references and words associated with communication. Negative and positive emotion did not vary significantly between the suicidal and non-suicidal groups.

Mulholland and Quinn (2013) venture to predict the suicidality of musicians through their song lyrics. Using a combination of syntactic, semantic and ngram features, they find a supervised classifier to be successful in detecting songs by a suicidal lyricist in 70% of the cases. First-person singular pronouns, concrete nouns and passive constructions are among the most informative features.

In electronic health records

Machine learning techniques have also been used to identify patients at high risk from suicidal behaviour, using the information contained in electronic health records (EHR). Tran et al. (2014) built a predictive model for 1 to 6 month suicide risk, using administrative and demographic data, information on prior self-harm episodes and mental and physical health diagnoses. In a study with 7 399 patients, the model's predictions were compared to clinicians' diagnoses, and found to be superior. This indicates that EHR data mining can be used to improve risk estimation of patients with potential suicidal behaviour.

In addition to the clinical codes and numerical data, EHRs also contain free text (e.g. admission notes and discharge summaries), a source of unstructured information that is harder to take advantage of in data mining applications. Haerian et al. (2012) explore the use of NLP techniques to extract structured output from EHR notes, and use it in combination with clinical codes to detect potential relationships between drugs (e.g. antidepressants) or psychosocial stressors (e.g. depression, eating disorders, domestic abuse) to the incidence of suicidality. Models that incorporated information from free text were found to have much higher predictive value than those that only included clinical codes.

In online media

Work on the automatic detection of suicidal content in online media is scarce. To the best of our knowledge, only two papers have been published on the topic.

Much like the objective in the current study, Huang et al. (2007) explore the possibility of mining social networking sites in the hope of identifying bloggers who are at risk of suicide, so that appropriate intervention can take place. They work on a corpus of crawled blog entries from MySpace.com users aged between 15 and 24 ($n = 4268$). A dictionary of suicide-related keywords and phrases was manually collected, with each entry receiving a weight between 1 and 20 based on 'suicide significance'. Using exact and relaxed pattern matching, each post is assigned a cumulative score, and bloggers are ranked by the average of their three highest-scoring posts. A qualitative analysis of the twenty highest-

ranking bloggers revealed that 35% of them exhibited signs of depression or suicidal tendencies. The 65% false positives contained lists of questions (e.g. *Has anyone close to you committed suicide?*), stories and biblical passages.

While it is apparent that this keyword-matching approach suffers from low precision, the data does not allow to measure recall, i.e. the number of actually suicidal bloggers that are missing from the results. The authors acknowledge that although the accuracy of the system is limited, it can serve as a proof of concept for the potential of more intricate text mining approaches to automatic suicidality detection.

A second study on user-generated content is that by Jashinsky et al. (2013). Like the previous study, it takes a keyword-based approach to detect at-risk content, on Twitter. Keywords were manually selected, by testing search terms linked to various risk factors and warning signs for suicide: depression and other psychological disorders, history of suicidality, drug abuse, antidepressants, self-harm, isolation, impulsiveness, bullying, family violence and firearms. A search term was retained if it appeared in tweets, accompanied by the expected suicide risk context, a condition that was manually verified in search samples. In a second step, a list of exclusion terms was identified for each search term, so as to remove tweets that were either jokes, nonpertinent or sarcastic in nature. This was done by manually inspecting sample tweets found with the search terms. An example of a search term is *cutting myself*, for which the exclusion terms *shaving*, *accidentally* and *slack* were identified.

The Twitter Streaming API was used to collect at-risk tweets for a three-month period. The search terms yielded almost 1.7 million tweets, 733 000 after applying the exclusion terms, and 38 000 for which the US state could be determined, based on GPS and user profile information. This corresponded to 28 000 users. A background corpus of random tweets was also collected and geolocated.

To validate the relevance of the results, all users were grouped by state, and the proportion of at-risk users versus background users was calculated. Proportions that departed from the expected (nation-wide) proportion were found to be strongly correlated to the actual age-adjusted state suicide rates. This indicates that Twitter (and possibly other social media content) may be viable for large-scale and real-time monitoring of suicide risk factors. A limitation of the study is that it may not be reliable on an atomic level, i.e. for specific Twitter users. While the high correlations suggest that the methodology yields a correct proportion of at-risk tweets, it is unclear how many of those tweets are false positives, and how many at-risk tweets are missing from the results. It may well be that precision or recall is low, but that it is consistently so across states.

The present study differs from the above work in that it investigates a suicidality detection approach that goes beyond keyword-based matching, instead using machine learning with many features. Furthermore, performance is evaluated on an atomic level, so as to determine the practical usability of the system for suicide prevention.

CHAPTER 3

Resources

In Chapter 1, we introduced the issue of suicide, the growing importance of the role of technology in suicidal behaviour, and the opportunities and risks presented by technology-based prevention efforts. Chapter 2 discussed the currently available work that is specifically concerned with suicidality in text, either through descriptive analysis or modelling approaches. More generally, it touched upon the application of natural language processing techniques for social purposes, such as forensics research.

This thesis is concerned with the task of detecting suicide-related material in user-generated content, and singling out messages that contain a high risk of suicide, which would make them a priority for review by a suicide prevention professional. The current chapter will discuss a number of resources that are indispensable for any such study, to wit:

- a text corpus of user-generated content, in which suicide-related material is present
- working definitions of whether a message can be considered suicide-related, and if so, whether it presents a high suicide risk
- guidelines that allow these working definitions to be applied consistently when annotating a corpus

An annotated corpus is a prerequisite for the evaluation of any automatic approach, because it provides a gold standard to which predictions can be compared. What is more, the experiments presented in this thesis are based on inductive learning techniques. These infer a prediction model from a set of training examples, which are taken from a corpus.

Section 3.1 introduces the various corpora that were used in this study: we first discuss the suicide-related corpora (3.1.1), then the reference corpora, which contain general social media content without a bias towards suicidal material (3.1.2), and finally, a number of corpora-related challenges are presented, along with proposed solutions (3.1.3). Section 3.2 presents the annotation guidelines that were developed for the task (3.2.1), the way in which annotation was carried out (3.2.2), the task definition (3.2.3), the results of an inter-annotator agreement study (3.2.4) and a number of statistics about the annotated corpus (3.2.5).

3.1 Corpora

For the experiments in this study, a corpus containing Dutch-language social media posts was needed. Ideally, one could use a substantial reference corpus on which previous research has been carried out, and annotate it for suicidality. This would, however, be impractical for two reasons. First, there is no guarantee that such a general corpus would contain enough suicidal material, if any, to be suitable for modelling suicidality with a machine learning approach. Second, even if the corpus were sufficiently large to contain a non-trivial amount of suicidal material, it would be infeasible to have it manually annotated, because the effort of sifting through irrelevant messages would considerably outweigh the actual annotation effort on positive instances, and thus, squander valuable annotator time.

We therefore opted for a two-pronged strategy for corpus collection. On the one hand, we obtained a corpus that would contain a high percentage of suicide-related messages, which was presented to the annotators. We also collected suicide-related text of other genres (such as web pages and chat logs), to be used as background material. On the other hand, a number of general corpora were used to offset the high number of positive instances in the annotated corpus. An important concern was that the format of these reference messages should differ as little as possible from the suicide-related material. Otherwise, reference messages would be too easily identifiable, and this would provide a machine learner with undue help in discarding them in the search for positive instances.

3.1.1 Suicide-related text corpora

The Netlog suicide corpus

The primary corpus used for suicide annotation consists of content from Netlog, an online social networking site based in Belgium, and owned by Massive Media Match NV. At the time of the corpus compilation, the platform was popular primarily amongst adolescents.

According to the Netlog website[1], its community counts over 96 million regularly active members, speaking 40 different languages. Members can create and maintain a personal profile page, called a *log*, where they can post blog entries, photos, videos, events, music playlists, etc. These are automatically shared with connected friends, and may be publicly accessible, depending on the user's privacy settings. Friends can post reactions on all content uploaded by a user.

Groups are pages that are not linked to any specific personal log, but to which members can subscribe. They cover a wide range of topics, and have linked forums on which members can discuss by means of forum posts. The groups feature was available on the platform at the time of data collection, but has since been discontinued.

Through the Belgian Centre for Suicide Prevention (CPZ[2]), we managed to obtain a collection of 1 380 posts from the Flemish section of Netlog with a high incidence of suicide-related content. The corpus was made in the fall of 2010, and consisted of 945 blog posts and 435 forum posts.

It is unclear what procedure had been followed to select these posts, although we can assume that the corpus is a compilation of messages that had been flagged as suicide-related, and messages matching a keyword search. Keywords that were likely used are the English term *suicide* and its translations in Dutch, namely *zelfmoord* and *zelfdoding*. These terms occur in 32, 1 203 and 55 posts, respectively. In 24 posts, more than one of the terms is present, and none is found in 95 posts. Inspection of these latter posts did not suggest that other search terms were used.

Statistics about these posts are presented in Section 3.2.5.

[1] http://nl.netlog.com/go/about, retrieved 10 April 2014
[2] http://www.preventiezelfdoding.be

Other suicide-related material

Apart from the Netlog corpus, other textual resources containing suicide-related material were obtained. These could be used as background material, for example to extract terms relevant to the domain (see 4.3.2), or to use as input for vector space models (see 4.4). One such resource consists of suicide-related pages crawled from the web, and is described in detail in Chapter 4, page 59.

A second resource is a collection of transcripts of chat conversations between people contacting the CPZ emergency chat service, and its trained volunteer responders. Due to its sensitive nature, this data is anonymized and temporarily stored for the purpose of volunteer evaluation and support. It was used for the extraction of relevant text passages, as discussed in Section 3.2 below. The collection contains a total of 290 conversations, consisting of 29 269 turns (average of 100.9 turns per conversation). The average turn length is 11.2 tokens (59.7 characters), and the total corpus size is 327 978 tokens (1 747 518 characters).

3.1.2 Reference corpora

Aside from corpora mainly consisting of suicide-related content, the experiments required a reference corpus with instances that had a low probability of being about suicide. This corpus should provide a large number of messages that can remain unannotated, under the assumption that it is safe to label them as negative instances for classification.

As mentioned above, care had to be taken to ensure that posts from the reference corpus were not easily distinguishable from those in the suicide corpus. We therefore used a data dump of 373 349 Netlog posts from the same period, on which no selection had been performed. Some cleanup was required (see 3.1.3), but the bulk of these messages were available for experiments.

In order to verify the assumption that all posts in the reference corpus could be labelled as negative (not suicide-related), we performed two checks on the data. First, we searched for the three keywords in the reference corpus. Whereas *zelfdoding* did not appear in any post, *zelfmoord* and *suicide* appeared in 42 and 10 posts, respectively. These posts were removed from the corpus.

Second, in order to account for less explicit suicide-related posts that did not contain any of the search terms, we performed a manual spot check on a sample of 1500 messages that had been randomly selected. Given that none of these contained suicidal material, we believe that it is liberal but justified to assume that there is no suicide-related content. Any positive instances in the reference

corpus would receive an erroneous gold standard label (a type II error or false negative), but their small number would not have a significant impact on overall scores.

3.1.3 Problems with the Netlog corpora

Both Netlog corpora were collected and stored by Netlog staff. There were a number of issues with the data that could have been avoided, but because we did not receive the data from Netlog directly, these problems could not be fixed at the root. We therefore had to resort to some workaround solutions, described below.

Missing line break information

In the Netlog suicide corpus, all newline characters had been removed. As a result, each post was stored in a single line, as shown in the following example:

(3) Tis weer hetzelfdedezelfde klojos waar je verliefd op wordZe gaan toch weer lopen met een ander meisjeJy krygt nooit die kansTzal altyd zo blyven Het mikpuntAlle gevoelens worden opgekropt [...] Blyf jy my altyd trouw...Die vraag blyft rondspoken Voor altyd

Some tokens, originally split by a line ending, are concatenated into one token. This is obvious in cases where a new line starts with a capital letter (e.g. *wordZe*, *meisjeJy*), or when punctuation is present (e.g. *trouw...Die*). However, there is no straightforward procedure to remedy these cases programmatically, because word-internal punctuation and capitalization can occur without warranting a line ending. Dots in abbreviations, for example, rarely serve as full stops. Capital letters do not occur exclusively in sentence-initial position, as they are frequently used for named entities. Randomly capitalized words are also quite common in user-generated content. When a sentence does not start with a capital letter, the word form of the resulting concatenated token (e.g. *hetzelfdedezelfde*) does not differ from regular tokens.

This issue affected a significant number of tokens, which are the starting point for many prediction features. Automatic sentence boundary detection would not be effective in addressing it, because it is not built to predict sentence boundaries inside tokens. We therefore manually reintroduced newline characters into the corpus.

A number of heuristic rules were defined to determine where a line break was originally present. Concatenated word forms, either with or without internal capitalization or punctuation, were deemed to have been split by a line break if the last constituent part was clearly the start of a new sentence or thought. Concatenated word forms that were the result of typing or spelling errors (space omissions) were not changed.

Newline characters were also inserted when two non-concatenated tokens were not likely to be part of the same sentence, but had no punctuation between them (e.g. *rondspoken Voor*). In such cases, the original text is assumed to have had a leading or trailing space around a line break. This is particularly frequent in posts with an apparent structure, such as lists or poems.

In the previous example, line breaks were inserted as follows:

(4) Tis weer hetzelfde
dezelfde klojos waar je verliefd op word
Ze gaan toch weer lopen met een ander meisje
Jy krygt nooit die kans
Tzal altyd zo blyven
Het mikpunt
Alle gevoelens worden opgekropt [...]
Blyf jy my altyd trouw...
Die vraag blyft rondspoken
Voor altyd

Badly encoded characters

The large Netlog reference corpus did contain line break information, but was poorly encoded. All posts were entirely composed of characters from the ASCII code page,which is limited to Latin letters without accents, numbers and a range of punctuation symbols and control codes. Any characters not available in ASCII were represented by a question mark. This includes, for example, all characters with accents, which could have been successfully encoded with an international character set such as UTF-8. Below are a few excerpts from affected posts.

(5) Toch een mooie maandag en hopelijk tot snel h?

(6) Mss nog k? op d manege , of we moetn k? aspreken .. xD

(7) oooh in Lier z? , Kont ge mij ni ff verwittige? :D

(8) Jet n? moksj?..

(9) H?nn?l?? ?? ??? !!!!

(10) Al? al?, niet tegenspreken h?! ;o

The replacement of special characters by question marks is problematic for a number of reasons. First, non-ASCII characters do occur in the small Netlog suicide corpus, and the presence of such a special character would therefore be a salient clue for a classifier to identify a post as pertaining to that corpus. Likewise, it changes tokens with special characters in the reference corpus into an easily distinguishable, and illegible, form (e.g. *H?nn?l??*). And finally, the replaced characters cannot be readily distinguished from valid question marks (e.g. *verwittige?*).

The first problem was addressed by replacing special characters in the Netlog suicide corpus by their closest resembling ASCII counterparts, essentially making both corpora ASCII-only. This was done using the Python unidecode package[3]. It is a package for transliterating Unicode characters into ASCII, in a way that approximates how a human with a US keyboard would map them, and also contains hand-tuned mappings to produce ASCII approximations for symbols and non-Latin alphabets. For languages of Western origin, it is said to perform between good and perfect (Burke 2001).

This approach effectively removes potentially useful information from the corpus. However, the usage of special characters is inconsistent in user-generated content. They may be omitted for ease of typing (e.g. *manege*), or used excessively (e.g. to add stress). It can therefore be argued that mapping all characters to their canonical form is a useful preprocessing step that abstracts away from unwanted variation.

To clean the unwarranted question marks in the reference corpus, we deemed it acceptable to filter out posts containing them, given the size of the corpus. Filtering out all posts containing a question mark would introduce a strong bias towards short posts (where the probability of encountering a specific character is lower), and it would also remove all the posts with valid questions in them.

We therefore used a series of conservative regular expressions. First, all occurrences of whitespace followed by *h?* or *h??* were replaced by *he* or *hee*, given the high number of posts that originally may have contained *hé*, *hè*, *héél*, etc. Next, occurrences of *?* followed by a lowercased character were filtered out (e.g. *H?nn?l??* from the examples above), as were single characters followed by *?* (e.g. *k?*, *z?* and *n?*). Finally, posts with *?* followed by a full stop or a comma, or a question mark at the start of a line, were discarded.

[3]https://pypi.python.org/pypi/Unidecode

These filters respectively matched 3 083, 22 748, 2 112, 5 285 and 2 782 posts. In total, 32 927 posts were removed, and 69 943 posts with question marks remained. Of the examples quoted above, only the first would not be removed.

In the Netlog suicide corpus, 36.1% of the posts contain question marks. In the reference corpus, this percentage dropped from 27.6% to 18.7% after cleanup. It should be noted, however, that the percentage before cleanup is artificially high because of invalid question marks.

3.2 Annotation

The aim of this thesis is to investigate the feasibility of automatic detection of suicide-related and high-risk posts in user-generated content. This filtering should allow prevention workers to monitor the so-called *firehose of social media* more effectively, and to respond quickly in cases where a response is deemed appropriate and necessary.

In this section, we propose a way to operationalize that task.

3.2.1 Annotation scheme

An annotation scheme had to be developed that would allow distinguishing between relevant and irrelevant posts, based on a working definition of suicide, discussed below. In order to subsequently determine which posts could be considered high-risk and should be flagged for review by a prevention worker, a structured annotation scheme was developed over a number of iterations, in close collaboration with CPZ.

The scheme, outlined in Figure 3.1, is based on five criteria that are commonly used for suicide threat assessment in prevention practice (Kerkhof and van Luyn 2010): the subject of the threat, the severity of suicide ideation, i.e. suicidal thoughts, the language used to describe it, and the presence or absence of risk factors and protective factors. These criteria are annotated in succession, although the scheme first discriminates between posts based on genre. At each level of annotation, the annotator can indicate uncertainty about a decision: the option *I do not know* is always available, but is omitted from Figure 3.1 and the discussion below, for brevity. Finally, the scheme covers the annotation of text passages that are salient for the task.

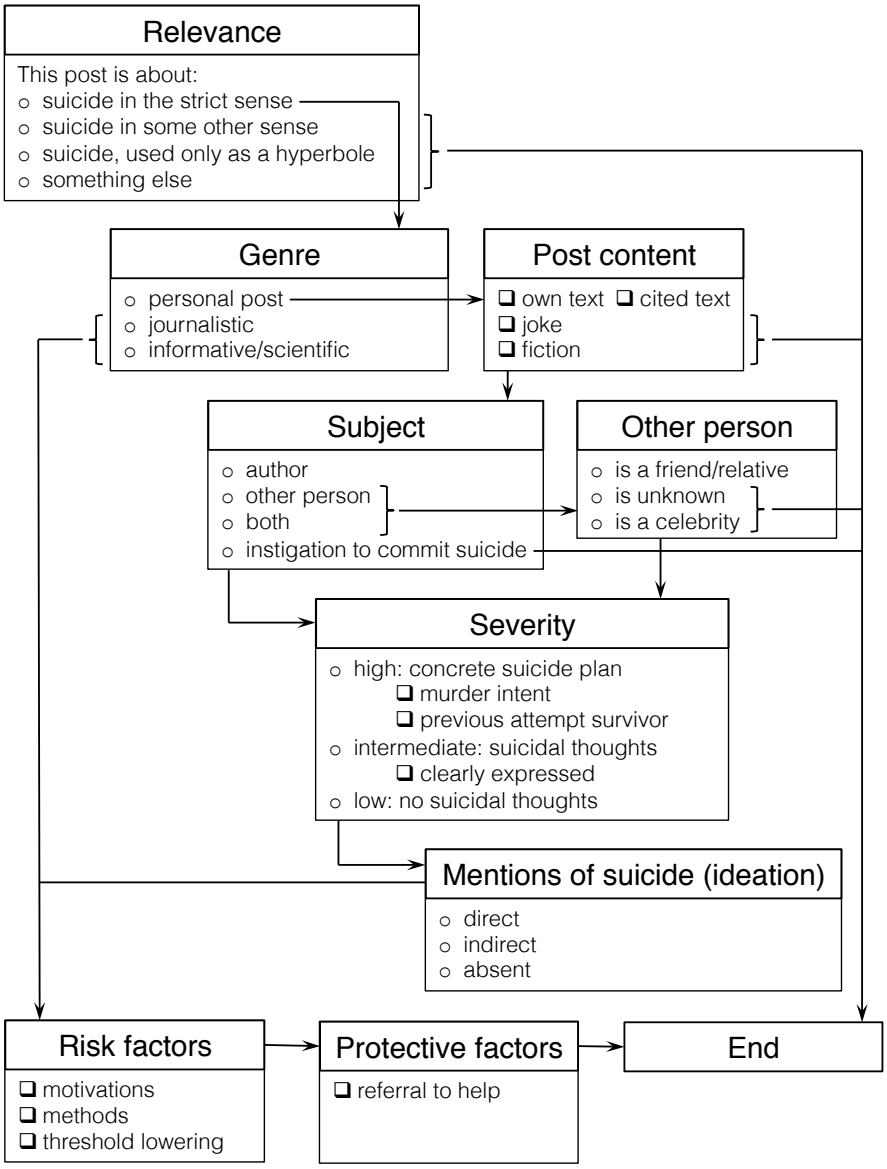

Figure 3.1: Schematic overview of the text level annotations. Round radio buttons indicate exclusive choices, square checkboxes indicate non-exclusive options.

Relevance

Suicidological research has been burdened with terminological ambiguity. The terms and definition used in the field, for purposes of diagnosis and medical classification, are many and varied (suicide, suicide attempt, parasuicide, suicidal behaviour, etc.). This is corroborated by a number of articles that sought to resolve this ambiguity (O'Carroll et al. 1996, Silverman et al. 2007a,b, De Leo et al. 2006).

We adopt two definitions proposed by De Leo et al. (2006), which synthesize previous definitions and are now widely accepted:

> **Suicide** is an act with fatal outcome, which the deceased, knowing or expecting a potentially fatal outcome, has initiated and carried out with the purpose of bringing about wanted changes.

> Non-fatal **suicidal behavior**, with or without injuries [can be defined as] a nonhabitual act with nonfatal outcome that the individual, expecting to, or taking the risk to die or to inflict bodily harm, initiated and carried out with the purpose of bringing about wanted changes.

For the sake of simplicity, we use the terms *suicide*, *suicide-related* and *suicidal* throughout this thesis to refer to both phenomena.

Technically, we consider a post to be relevant (1) if it is suspected to contain textual evidence of *suicidal behaviour*, where prevention efforts can still have an effect, or (2) if it contains mentions of *suicide*, either because of suicidal thoughts, or through accounts of suicides by others. These posts are said to be about *suicide in the strict sense*.

Conversely, irrelevant posts would be those that (1) have nothing to do with suicide, (2) use suicidal imagery hyperbolically (e.g. as a figure of speech), or (3) treat the concept of suicide according to some other definition. These posts are not considered for further annotation. Below are two examples of each case.

(11) NL: Hallo, allemaal. Iedereen heeft het wellicht gehoord. dat vorige week een vrouw haar vijf kinderen heeft vermoord. Ik vind het uiteraard niet kunnen! Maar we weten ook niet wat er achter de muren heeft afgespeelt *[sic]* ...

EN: Hi, all. I suppose everybody heard. that a woman killed her five

children last week. I think that's terrible! But we don't know what really happened there...

(12) NL: Zeer opvallend zijn de achterwaarts scharnierende, halve achterdeuren (ook wel 'suicide doors' genoemd), die de twee achterstoelen ontsluiten.

EN: Strikingly, the back doors (also called 'suicide doors') swing open backwards, and allow access to the two back seats.

(13) NL: weggaan van u zou zelfmoord zijn, ik kan nie leven zonder u.

EN: leaving you would be suicide, I can't live without you.

(14) NL: Maar ik denk niet dat we nu collectief zelfmoord moeten plegen. Ik heb het gevoel dat Club de laatste weken progressie maakt, individueel en qua spelsysteem.

EN: But I do not think we should commit mass suicide. I feel that Club has made good progress in the past weeks, both individually and tactically.

(15) NL: Aan deze zelfmoordenaars beloofd men dat zij in het paradijs alles kunnen krijgen waarvan ze op aarde enkel hebben kunnen dromen... WEES DAN EERLIJK! IS JE OPBLAZEN DAN NIET EEN LOGISCHE WENS!!!!!!?

EN: They promise the suicide attackers that in paradise, all their dreams will come true... SO BE HONEST! ISN'T IT LOGICAL THAT THEY WANT TO BLOW THEMSELVES UP!!!!!!?

(16) NL: Een andere houding van de N-VA zou politieke zelfmoord betekenen.

EN: Any other position would be political suicide for N-VA.

Genre

For relevant posts, annotators have to distinguish between posts that have been written from a personal perspective (for an audience of friends or with personal opinions), and those that have been written for a broader audience (i.e. journalistic, informative or scientific texts). For the latter category, we skip to the annotation of risk and protective factors, which are relevant in determining whether the media guidelines have been followed.

For personal posts, we mark whether they contain personally written text, citations (e.g. lyrics of a song, text quoted from another user), jokes or other fictional accounts of suicide. In case the references to suicide are all fictional, the post is not annotated further.

Subject

Next, we determine the subject of the suicide mentions or suspicions. This can either be the author of the post, some other person, or both. For other persons, it is indicated whether this is a friend or relative (known to the author), some unknown or generic person (e.g. when discussing suicide in general), or a celebrity. If the post is not about the author or some known peer, annotation ends.

The particular case where the post author instigates some other person to commit suicide is annotated separately, after which annotation ends as well. These cases can be of particular interest for the study of online bullying behaviour (*cyberbullying*).

Severity

We have established that the post is written from a personal perspective, and either deals directly with a non-fictional case of suicide (ideation) by the author or a known peer, or contains indications to that effect, as inferred by the annotator.

The severity of the suicide threat (for the annotated subject) is now analysed. When there are no suicidal thoughts, severity is *low*. A typical example would be a descriptive post about the suicide of some other person. If the subject has expressed, or is assessed to have, suicidal thoughts, the severity is considered to be *intermediate*. In cases where the suicidal thoughts have developed into a death wish and produced a suicide plan (the subject has made concrete decisions about time, place and method), severity is annotated as *high*. Two cases of aggravating circumstance are annotated: whether there is cause to believe there is a homicide threat, and whether the subject is a survivor of previous suicide attempts.

Mentions of suicide ideation

This criterion describes the specificity of the language used to mention suicide, as pertaining to one of three levels:

- direct, e.g. *NL: nu maak ik er een eind aan / EN: I will end this now*, *NL: Ik denk nog elke dag aan zelfmoord, ik wens mijzelf nog steeds dood / EN: I still think about suicide every day, I still wish I were dead*)
- indirect, e.g. *NL: zou het ni beter zijn als ik nu zou gaan ? / EN: wouldn't it be better if I went now?*, *NL: Als het me allemaal te veel wordt, heb ik de neiging om mezelf iets aan te doen/ EN: When it all becomes too much, I feel like hurting myself*
- absent

Risk factors and protective factors

Finally, the annotator can indicate whether any risk or protective factors are present in the text. Kerkhof and Van Heeringen (2000) describe the suicidal process as an interaction between risk factors, which may cause a person to be suicidal, and protective factors, which act as buffers against those risks.

Three types of risk factors are annotated.

- *Motivations* for suicide may cause a person to become suicidal (e.g. loss, medical conditions, mental disorders or substance abuse)
- Mentions or descriptions of suicide *methods*
- Factors that *lower the threshold* for committing suicide. As opposed to motivations, these factors do not cause suicidality, but they can act as a catalyst for suicide (e.g. previous attempts or the availability of a method).

Examples of protective factors are social network, professional help, physical wellbeing, religion, fear of pain or death, consequences for relatives, etc.

The annotation of risk and protective factors in published texts (such as newspaper articles) is relevant to check their compliance with the media guidelines proposed by CPZ. These stipulate how suicide-related news should be presented, i.e. without explicitly mentioning risk factors so as to avoid copycat behaviour, and with a reference to help outlets.

Text span annotations

In addition to the text level annotations, annotators are asked to mark relevant passages in the text itself. We distinguish five types of text passages:

- **Alarming** situations or feelings that are described in the post, that do not necessarily indicate suicidality (e.g. *NL: vroeger werd ik gepest / EN: I was bullied when I was younger* or *NL: ik ben ten einde raad / EN: I am at my wits' end*).
- Phrases that clearly indicate **suicidal** thoughts (e.g. *NL: Waarom zou ik nog leven? / EN: Why would I continue living?* or *NL: kwil eiglijk gwn doodgaan / EN: I just want to die*). These phrases are used to build a lexicon of suicide-related terms (see 4.3).
- *Citation*, a section of the post that was not originally written by the post author, such as the lyrics of a song, or text quoted from another user.
- *Risk factor*, as described above. The type of risk factor is indicated.
- *Protective factor*, as described above.

3.2.2 Annotation implementation

All posts in the Netlog suicide and chat transcript corpora were manually annotated. Annotation was carried out by two volunteers, three members of staff and one intern at CPZ, over the course of eight months. The bulk of the annotations was done by the volunteers, who could work from home, with staff working mainly on the annotation of text passages in the chat transcripts.

The tool of choice for annotation was BRAT[4], the brat rapid annotation tool (Stenetorp et al. 2012). It takes UTF8-encoded text files as input, and stores the annotations in a proprietary standoff format. The features of BRAT met most of our requirements:

- BRAT has an intuitive graphical user interface, which reduces annotator training time and improves the speed of annotation.
- It is a web-based application that runs in a browser. BRAT therefore does not require local installation, and is platform-independent on the client side. It allows online work from any location, which was a necessity for the

[4]Available at http://brat.nlplab.org/

volunteer annotators working from home. The client-server architecture also simplifies file management, because distribution and back-up of files and annotations is centralized on the server.

- BRAT is designed for structured annotation of text spans (entities), their attributes, relations between them, and event annotations, which link together any number of other annotations participating in specific roles.
- It supports character level annotation. The granularity of an annotation tool is defined by how units for annotation are represented internally. Some tools only allow annotation on the token level. This would be too restrictive for our case, because user-generated content cannot always be properly tokenized without normalization.
- The program is released under an open-source, MIT-compatible licence, its code is well-documented, and there is an active community of developers and users on the brat-users mailing list[5].

An important feature missing from BRAT is support for text level annotations. These are annotations that are not linked to a specific span of characters or tokens in the text, but that consider the text as a whole as the unit for annotation. This could be used to provide metadata about the text itself, such as the suicide relevance annotations in our case.

This limitation was addressed in the work of Naert (2013). It extends BRAT version 1.3 in a number of ways, providing improved caching, visibility controls, FoLiA[6] support, and two extensions that are especially relevant for our annotation task:

- Support for text level annotations, which can be freetext or structured. Structured annotation flow is defined in the configuration file, with sets of possible options to choose from. Each choice either points to a next set of options to be annotated, or concludes and stores the annotation. The user interface presents these sets in order, and allows navigation. When reviewing existing text level annotations, the previously selected option is highlighted (see Figure 3.3). If a choice is edited, incompatible downstream annotations are removed.
- Validation, allowing an annotator to have a document checked against a number of predefined rules (e.g. *Is the Type attribute set for every Risk Factor annotation?*, or in the case of relation annotations, *Are there no orphaned entities?*). Annotations that violate a rule are visually flagged

[5] https://groups.google.com/forum/#!forum/brat-users
[6] http://proycon.github.io/folia/

with a red glow, and error messages can be consulted in the annotation overlay. Validation thus provides immediate and user-friendly feedback that can help avoid easily preventable oversights and errors.

The extended version is currently being incorporated into the main BRAT codebase, and was used for the annotation by CPZ.

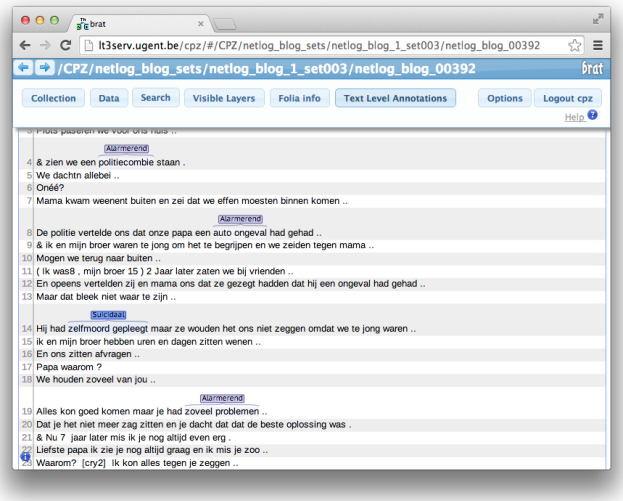

Figure 3.2: Main view of the BRAT annotation interface, with visible menubar.

The main view of the BRAT graphical environment is shown in Figure 3.2. Most of the browser window is used to display the text and its text span annotations. The collapsible menu bar at the top of the screen provides access to a file browser, the text level annotations panel (Figure 3.3), options, etc. In the panel in Figure 3.4, the entity type and attributes can be set for text span *Niemand mag jou* (*EN: Nobody likes you*).

3.2.3 Task definition

The annotations described above provide an abundance of information, from which a variety of tasks could be derived. This thesis is focused on two specific use cases: (1) the detection of suicide-related posts, which will be referred to as the *relevance task*, and (2) the detection of posts presenting a high suicide threat, which should receive priority attention from suicide prevention workers, henceforth called the *severity task*.

Figure 3.3: Panel for structured text level annotation in BRAT.

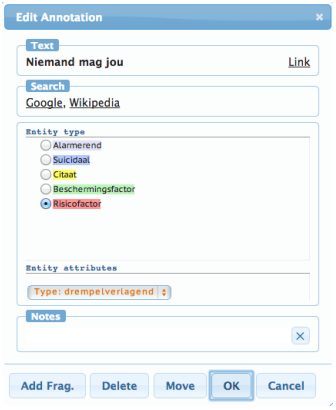

Figure 3.4: Panel for text span annotation in BRAT.

Based on the annotation scheme, the two tasks are operationalized as follows.

For the relevance task, posts that receive a *Relevance: suicide in the strict sense* annotation are considered positive instances. All other posts are negative instances.

Positive instances for the severity task are the posts that have a *Severity: high* or *Severity: intermediate* annotation. This corresponds to the set of personally written, non-fictional posts that contain evidence (as per the annotator's judgment) that the post author or a known peer has suicidal thoughts and/or a suicide plan.

Since all its positive instances are, by definition, also positive instances of the relevance task, the severity task can be rephrased as the task of reducing the set of relevant posts to its high-risk subset, as discussed in Section 5.4.

3.2.4 Inter-annotator agreement

In order to assess the reliability of the annotations, and whether the guidelines allowed the task to be carried out consistently, an inter-annotator agreement experiment was set up. A set of one hundred posts was collected by sampling 40 posts from the Netlog suicide corpus and 60 posts from the Netlog reference corpus. Three annotators at CPZ (one volunteer, one intern and one member of staff) were asked to annotate this set, independently from each other.

All annotations were converted to the class labels for the relevance and severity tasks, in order to test annotation reliability specifically for these tasks. We calculate pairwise agreement, and average agreement over all pairs, using three statistics:

- *Accuracy*, the percentage of cases on which the annotators agree.
- *F-score* on the positive class, calculated by taking one annotator as the gold standard and scoring the annotations of the other for precision and recall. This yields the same results as averaging the precision or the recall scores of both annotators, based on the other as gold standard.
- *Cohen's Kappa* statistic, or κ, a widespread measure in the field of computational linguistics to evaluate agreement on labeling tasks (Carletta 1996). It normalizes for the amount of expected chance agreement: when there is no agreement other than that which would be expected by chance, κ is zero. When there is total agreement, κ is one.

The pairwise and average scores are presented in Table 3.1

		A1-A2	**A1-A3**	**A2-A3**	**Average**
Relevance	Accuracy	0.95	0.94	0.93	0.94
	F-score	0.92	0.91	0.89	0.91
	κ	0.88	0.86	0.84	0.86
Severity	Accuracy	0.92	0.94	0.90	0.92
	F-score	0.73	0.67	0.69	0.70
	κ	0.65	0.59	0.50	0.58

Table 3.1: Inter-annotator agreement scores for the relevance and severity tasks.

From the accuracy scores, we can infer that on average, annotators disagree on 6 in 100 posts for relevance, and 8 in 100 posts for severity. The F-scores are of particular interest for comparison to classifier performance, which is also typically measured with F-score, as discussed in detail in Section 5.3. It provides a rough estimate of the difficulty of the task for humans, and could therefore be viewed as a ceiling for performance of automatic classifiers, which have to infer their model from imperfect human annotations. However, the score is tied to this particular dataset, so direct comparisons can only be meaningful when the same 100 instances are automatically classified.

The κ scores are best suited for evaluating the inter-annotator agreement. The average κ of 0.86 for the relevance task can be interpreted as good reliability ($\kappa > 0.8$). For the severity task, on the other hand, κ is moderately low at 0.58

on average. This can be explained by the number of choices annotators have to make before they reach the severity annotation, as errors percolate downstream.

A qualitative analysis of the disagreements reveals that there are no posts where all annotator pairs disagree on the *Relevance* annotation, for which there are four possible values. There are nine posts on which two annotators agree and one annotator disagrees, yielding 18 cases of pairwise disagreement (hence six on average for the three pairs). Three of these nine disagreements can be attributed to an avoidable error on the part of the annotator, six are caused by confusion. The confusion is between *suicide in the strict sense* and either *suicide, used only as a hyperbole* (4 cases) or *not about suicide* (2 cases). An example of the latter shows that relevance annotation can indeed be subject to interpretation.

(17) NL: [...] ik weet dat je weg wilde gaan en nu je het toch hebt gedaan wil ik je nog zeggen
had ik nu maar nagedacht en je vaker opgevrolijkt
had ik je nu maar blij gemaakt
maar ik weet dat ondanks alles wat ik gedaan zou kunnen hebben
je keuze vaststond lieve schat en dat je blij bent weg te zijn [...]

EN: [...] I know you wanted to go and now that you have done it, I want to tell you
I wish I had been mindful and cheered you up more often
I wish I had brought you joy
but I know that in spite of everything I could have done
you had made your choice, dear, and you're glad to be gone [...]

The 24 cases of pairwise disagreement on the severity labels correspond to 14 effective disagreements. Three of those are caused by disagreement on the relevance level. It should be noted that not all of the relevance disagreements discussed above entail disagreement on the severity task label as well, because negative severity labels will still match if one annotator deems a post irrelevant, and the other deems it relevant but not high-risk. Four cases of disagreement stem from avoidable errors, i.e. cases where one particular annotation choice is clearly correct, but was not selected by one of the annotators. Finally, seven cases are caused by confusion: five on the *Subject* level and two on the *Severity* level. No confusion was found with regard to *Genre*. Confusion about the subject of the threat can often be explained by the ambiguous use of third person subjects, such as in the example below. Some annotators consider these posts to be about some generic person, and therefore label them low-risk, while others interpret them as veiled expressions of suicide ideation by the author.

(18) NL: [...] zo is het leven, en elke mens vecht ervoor
maar er zijn mense die er nimeer tegen kunnen en die moeten gaan !
en dan is er 1 uitweg
ZELFMOORD dat angstig woord !
maar de enige uitweg naar het afscheid vn het leven wr ze nimr vr wiln vechtn!
zo is het leven.. en zo zal het altijd blijven !

EN: [...] such is life, and everyone fights for it
but some people can't cope anymore and they have to go !
and then there is 1 exit
SUICIDE that terrifying word !
but the only exit to part with a life they don't want to fight for anymore!
such is life.. and so it will forever remain !

Clarity about the subject of a potential threat does not preclude confusion about its severity. Annotators may need additional information to judge whether suicidal thoughts are in play, e.g. because of vagueness, or because of limitations of the medium (which is written and one-directional).

(19) NL: zelfmoord gedachtn =) [innocent]
en goesting om te dn
gwn alles zit tgn bij mij
en waarom edde na gevoeles vr iemand as ge toch gn kopl word:D

EN: suicide thoughts =) [innocent]
and I want to do it
just about everything goes wrong
and why do I have feelings for someone if we're not going to be together anyway:D

In summary, we believe the inter-annotator agreement study shows that the annotation guidelines allow reliable annotation for relevance, and are not the main cause of confusion for severity annotation. The qualitative analysis demonstrates the ambiguity inherent to the task, and to the medium: there are no infallible protocols for diagnosing suicide ideation (regardless of medium), and even if there were, the information that can be derived from a single social media message is likely too limited. The standard approach of CPZ towards ambiguity was therefore followed, namely to annotate pessimistically.

3.2.5 Annotation statistics

In this section, we present a number of statistics of the annotated data. After removal of duplicates and foreign-language posts, the Netlog suicide corpus consisted of 1 040 posts (down from 1 380). As detailed in Table 3.2, the average post contained 7.9 lines, 121 tokens, and 697 characters, although we notice considerable deviation from these averages, with posts as short as 4 tokens. As is typical for social media content, overall post length is relatively short, but not as short as content from specific microblogging platforms such as Twitter, which imposes a character limit of 140.

Length	**Average**	**SD**	**Min**	**Max**
Lines	7.87	10.96	1	122
Tokens	121.20	77.71	4	322
Characters	697.15	418.99	51	1650

Table 3.2: Post length in the Netlog suicide corpus (mean, standard deviation, minimum, maximum), in terms of lines, tokens and characters.

All posts were annotated and double-checked. Out of 1 040 posts, 82% ($n = 851$) are about suicide in the strict sense, 2% about suicide in some other sense, 12% use the topic conditionally or hypothetically and 5% are entirely unrelated to suicide. Following the definition for the relevance task, the annotated corpus therefore contains 851 relevant and 189 irrelevant posts. Since the majority of these irrelevant posts do contain references to suicide, distinguishing them from relevant posts is not a simple matter of keyword matching.

Relevant posts were further annotated as follows (we refer to page 31 for the schematic overview of the annotation scheme, summarized in Figure 3.5): 579 posts are of a personal nature, 235 journalistic and 37 informative (68%, 28% and 4%, respectively). The personal posts contain 52 fictitious accounts and 27 jokes about suicide. The vast majority (96%) of the remaining 500 personal posts was written entirely by the author, as opposed to 18 posts that contain excerpts from other blogs, news coverage, poetry or song lyrics.

The 500 non-fictitious personal posts about suicide were annotated with regard to the subject of the suicide-related content. 49% of the posts have the author as a subject, 60% some other person; both the author and some other person are the subject in 54 posts (11%). Nine posts (2%) contain an instigation to commit suicide. Of the 299 posts about some other person, 134 (45%) are about a personal acquaintance of the author, 27 (9%) about a celebrity, and 138 (46%) about suicide in general, without reference to any specific subject.

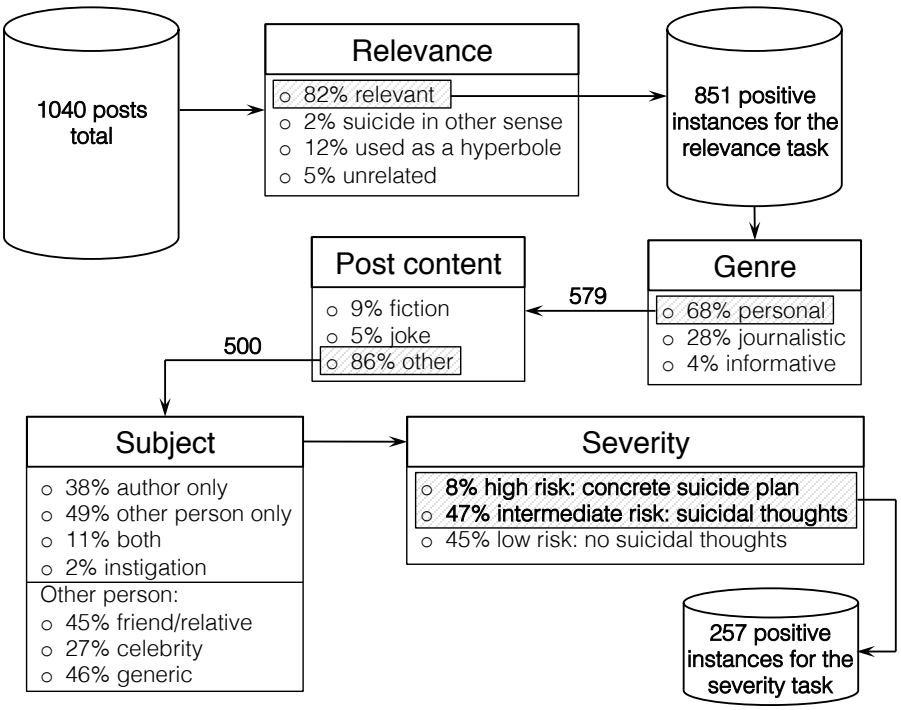

Figure 3.5: Breakdown of annotations per category.

In terms of severity, the corpus contains 39 high-threat posts where the subject is believed to have a suicide plan (8%), 218 posts with intermediate risk (suicidal thoughts, 47%), and 207 posts without indications of suicide ideation (45%). For the severity task, the posts with high and intermediate risk are pooled together, resulting in 257 severe posts (25% of the corpus).

Suicide is referred to explicitly in 91% of the posts, and implicitly in 9%. The high percentage of explicit posts can be attributed to the corpus collection process, which was partly based on keywords. However, those keywords did not necessarily appear in the contexts that made a message severe, nor were all messages containing a keyword considered relevant. Nevertheless, the low number of truly implicit posts related to suicide is a limitation of the corpus. We anticipate that the detection system resulting from this study may be instrumental in collecting additional material, both implicit and explicit.

Risk factors are present in 32% of the posts, most of which severe. Posts that contain risk factors usually include motivations (72%). Methods are mentioned

in 48%, and threshold-lowering factors are rare (28%). Protective factors occur in a mere 8% of the posts.

3.3 Summary

In this chapter, a number of resources were introduced that are indispensable for the study of automatic suicidality detection in social media.

First, we obtained and cleaned a large collection of user-generated content from the social networking site Netlog, some of which was suicide-related. Acquiring such suicide-related material presented an impasse: given its scarcity, manual collection would be overly time-consuming, and no automatic detection approaches currently exist. The collection method was therefore based on a keyword search, complemented with posts that had been manually selected. Additionally, a highly topical corpus of emergency chat transcripts was made available as background material.

Second, we defined sound working definitions of what should be detected in the relevance and severity classification tasks, based on a newly developed annotation scheme. The scheme is grounded in suicide prevention practice, and allows consistent annotation, as borne out by the inter-annotator agreement study.

Lastly, the annotation scheme was applied to a section of the corpus, so as to create a gold standard dataset for the inductive modelling and evaluation of the tasks.

In the following chapter, we continue with a discussion of information sources, another essential component for machine learning.

CHAPTER 4

Information sources

In supervised learning, a machine learning algorithm is given a set of training instances, from which it is to infer a model to predict a desired output for unseen instances. The desired output for a classification problem is a choice from a finite set of classes, e.g. *0* or *1* for binary classification. The objects for classification, called *instances*, are represented as a vector of attribute/value pairs, called *features*. These contain possibly disambiguating information for the classifier. For training instances, the correct class label is provided as well.

In our case, the object for classification is a social media message, and the desired output is a binary class label denoting membership of the *relevant* or *severe* subset, depending on the task. The messages are encoded as vectors of, ideally, highly informative features that allow the classifier to make accurate predictions. Features are said to be *salient* when they provide support to discriminate between classes. Conversely, inaccurate or irrelevant features should be avoided, because they can confuse the learner. Another aim of feature engineering is to provide abstraction, as this may do away with idiosyncrasies present in the training examples, so that the learning algorithm can more reasonably generalize to unseen instances. In addition, it can reduce vector sparsity, as multiple similar features may be collapsed into a more general abstraction.

This chapter deals with the problem of converting the input, a string of text, into a vector representation that exposes salient characteristics of that text to the classifier. We first discuss the preprocessing of the data in Section 4.1, and then introduce the features that will be used throughout the experiments: bag-of-words features (4.2), lexicon-based features (4.3), features derived from topic models (4.4) and some miscellaneous other features (4.5).

4.1 Preprocessing

The input to our system is a corpus of raw strings of text. As described in Section 3.1.3, this text had been converted with unidecode, so as to only contain ASCII characters.

Some features are calculated directly on the original text, but for most features, we use a version of the text that had first been preprocessed. Shallow linguistic analysis was performed with Pattern, a Python package for web mining, NLP and machine learning (De Smedt and Daelemans 2012a), and consisted of three tasks:

1. **Tokenization** is the task of splitting off punctuation from words, so that all tokens are separated by whitespace. Tokens can be words, numbers, punctuation marks, etc. Tokenization should leave certain sequences unchanged, such as acronyms, abbreviations (where the periods do not have to be separated from the preceding letters), or apostrophes used as a genitive mark.

2. During **part-of-speech tagging**, a grammatical category or *part-of-speech* code is assigned to each token. Pattern outputs tags from the Penn Treebank tag set, such as *NN* for singular nouns and *JJ* for adjectives.

3. The process of **lemmatization** generates the base form or *lemma* for each token. For verbs, this base form is the bare infinitive, whereas for the other grammatical categories, the base form corresponds to the *stem* of the word, i.e. the word without inflectional endings. Lemmatization therefore requires the part-of-speech information from the previous step, e.g. to disambiguate word forms with multiple lemmas depending on their grammatical category. This is illustrated in the English word form *meeting*, which can be a verb (lemma *meet*), or a noun (lemma *meeting*).

The Pattern parser is lightweight and optimized for speed, and is therefore well-suited for processing large amounts of data. We compared its output to that of

two other shallow parsers for Dutch: Frog (van den Bosch et al. 2007) and LeTs Preprocess (Van de Kauter et al. 2013). All parsers had difficulty with noisy input, and the results obtained with Pattern were only slightly less accurate. We concluded that this was an acceptable trade-off for its higher speed.

The tests with Pattern showed that it, predictably, performed badly on the noisy content in our corpus. We therefore cleaned and pretokenized the text, to improve tokenization and lemmatization performance, and to remove uninformative variation that would impede abstraction. An example of a text excerpt before and after cleaning and pretokenization is shown below. The effect of cleaning on the Pattern output can be found in Table 4.1.

(20) *Original:* [b]De eerste "date".[\b]Je zwaaide naar me...en ik WIST gwn dawe beste vriendinnen gingen worde:)

(21) *Cleaned:* De eerste date . Je zwaaide naar me ... en ik wist gwn dawe beste vriendinnen gingen worde

As a first step, regular expressions were used to remove or replace a number of occurrences that are typical for our corpus, or for user-generated content in general:

- Hyperlinks are replaced by the token *URL*.
- Character escape sequences that persisted in the text, possibly as an artefact of copy-paste operations by Netlog users, were restored to their corresponding characters, e.g. `&` to `&` or `\"` to `"`.
- In the corpora, tags of the form `[sad]` or `[\b]` were present to insert e.g. animated emoticon images or to add markup. These were removed.
- A wide range of plaintext emoticons were also removed. The prototypical smiley face emoticon consists of a colon followed by a right parenthesis (`:)`), but many productive variations were considered as well (e.g. `B-)`, `;=]`, `<3` or `^_^`).
- All double quote characters were removed, because users may add them (e.g. for emphasis) in the middle of significant word sequences.

As shown in Table 4.1, the presence of the `[b]` and `[\b]` tags is detrimental for tokenization accuracy. The emoticon and double quotes are removed.

Next, undue capitalization was addressed. All sequences of more than one contiguous capital letters were lowercased. This left words starting with a capital

Original		Cleaned	
Token	**Lemma**	**Token**	**Lemma**
[	[		
b]De	b]de	De	de
eerste	eerste	eerste	eerste
“	“		
date”.[\b]Je	date”.[\b]je	date	date
		.	.
		Je	je
zwaaide	zwaaien	zwaaide	zwaaien
naar	naar	naar	naar
me...en	me..	me	me
		...	...
		en	en
ik	ik	ik	ik
WIST	wist	wist	weten
gwn	gwn	gwn	gwn
dawe	daaw	dawe	daaw
beste	best	beste	best
vriendinnen	vriendin	vriendinnen	vriendin
gingen	gaan	gingen	gaan
worde	worren	worde	worren
:)	:)		

Table 4.1: Pattern tokenization and lemmatization output for an example sentence before and after cleaning and pretokenization.

intact. In the example, *WIST* is incorrectly lemmatized to *wist*, because the capitalized form is considered a noun. After lowercasing, the correct lemma *weten* is given.

Finally, the text was pretokenized with a number of rules. This step introduced spaces in locations where the tokenizer was found to produce errors, namely when words are written together without a space after a punctuation mark (e.g. *hebben,maar*, or *date.Je* and *me...en* from the example). The rules split up tokens that contain internal punctuation, except when both characters around the punctuation are uppercased (as may be the case in abbreviations). Redundant whitespace, such as multiple spaces or line break characters, was reduced to a single instance.

In what follows, we describe the features as having been calculated on a specific text layer.

- The *original* text layer contains the raw string before preprocessing, so that the full extent of orthographic variation, capitalization, tags, emoticons, etc. is still available.
- The *clean* layer contains a cleaned, tokenized version with lemma information.
- The *last* layer only contains the last 10 tokens of the cleaned text. The intuition behind this layer is that it may contain a summary of the emotions or topics discussed, which is not diluted by what comes before. The same rationale has been put forward in document summarization and sentiment analysis approaches (Liu 2010).

In Chapter 8, the clean layer is replaced with a normalized one, to study the effect of applying automatic normalization of noisy text on vector sparsity and classification performance. Examples of noisy tokens, typical for user-generated content, are *gwn* (*gewoon*; EN: *just*), *dawe* (*dat we*; EN: *that we*) and *worde* (*worden*; EN: *become*), which all cannot be lemmatized correctly.

4.2 Bag-of-words features

A bag-of-words model is a simplified representation of a text, in which the presence (or frequency) of its words is coded as features in a vector. It therefore considers a text as an unordered set (or *bag*) of words, disregarding word order and grammar. A corpus represented as bag-of-words vectors is an example of

a *vector space model*. Bag-of-words features have proven very successful in the domains of information retrieval and text classification (Manning et al. 2008).

In bag-of-words representations, each feature corresponds to a single word found in the training corpus, usually with case and punctuation removed. Words with a high frequency in the corpus are typically filtered out, as well as words appearing in a list of stop words (functional or connective words that are assumed to have no information content). Hapaxes and other low-frequency words are often removed, because they are unlikely to generalize well to unseen documents. Additionally, stemming or lemmatization can be applied. Stems and lemmas have the advantage over raw tokens that they are statistically more independent, since several morphological variants of a word are mapped to a common base form. This further reduces feature set size and possibly improves recall.

These filtering methods are standard practice in text classification systems (Scott and Matwin 1999). However, the universality of their effectiveness has been questioned. Riloff (1995) argues that some types of words, which would normally be filtered out or merged, play an important role in making certain discriminations. For example, similar expressions containing different prepositions and auxiliary verbs were found to behave very differently, and singular and plural nouns produced dramatically different text classification results.

Furthermore, the tasks at hand cannot be directly compared to text categorization problems. The 'topic' of suicide may not always be readily apparent, and therefore, we should not discard words that may have to do more with style than with topic. For these reasons, no filtering based on word category or frequency was done, as this might remove potentially salient features for our task. As a result, our feature space is large. In Chapter 6, we describe various techniques to reduce its size.

Bag-of-words feature values can be binary, for presence of a word, or a frequency count. Frequency counts can be weighted and normalized, e.g. using the *term frequency-inverse document frequency* (*tf-idf*) statistic, which estimates how specific and important a term is to a document, compared to a corpus. Given a document collection D, a word w, and an individual document d *in* D, tf-idf weight $W_{w,d}$ is calculated as follows:

$$W_{w,d} = f_{w,d} \cdot \log(|D|/f_{w,D}) \tag{4.1}$$

where $f_{w,d}$ equals the number of times w appears in d, $|D|$ is the size of the corpus and $f_{w,D}$ equals the number of documents in D in which w appears (Berger et al. 2000).

Tf-idf weighting is commonplace in text categorization, but it has drawbacks, as noted for example by Salton and Buckley (1988), who found that binary weights without normalization were to be preferred on short documents. Also, in our application we are not so much trying to find the most frequent or document-specific terms or topics (as promoted by tf-idf), but rather the presence of one topic, however small. All bag-of-words features are therefore coded as unweighted, binary values.

The principle of vector space models of words or lemmas can equally be applied to sequences of items, called *n-grams*. As opposed to *unigrams*, n-grams such as *bigrams* or *trigrams* can be effective in capturing local context or salient collocations.

Likewise, bags-of-n-grams can be constructed from character sequences. Characters have proven their usefulness in fields such as topic detection (Clement and Sharp 2003), where they improve reliability on short text fragments. We also expect them to be more robust to noise, like orthographic variation, than token-based representations.

Given the above considerations, the following bag-of-words features are used in the experiments:

- Word unigrams (W1), bigrams (W2) and trigrams (W3), taken from the clean layer.
- Lemma unigrams (LEM1), bigrams (LEM2) and trigrams (LEM3), clean layer.
- Lemma character bigrams (LCH2), trigrams (LCH3) and fourgrams (LCH4), clean layer.
- Word character bigrams (WCH2), trigrams (WCH3) and fourgrams (WCH4), taken from the original layer. Unlike the cleaner and more abstract lemma character n-grams, these n-grams have access to emoticons, tags, etc.

Case is removed in the clean layer. In both the original and the clean layers, punctuation is retained.

4.3 Lexicon-based features

4.3.1 Polarity lexicons

In computational linguistics, a lexicon is an inventory of lexical items, containing the entire vocabulary of a language or a subset thereof. Most current methods for automatic subjectivity analysis, for example, rely on lists of words with polarity, i.e. words that are usually associated with positive or negative sentiments or opinions (Liu 2010). Lexicons may be flat lists, i.e. simple collections of salient words, but the entries in a subjectivity lexicon are typically enriched with meta-information like polarity and intensity scores.

Because we suspect that negative (or lack of positive) polarity in a post may be correlated to suicidality, we implemented polarity features based on two subjectivity lexicons available for Dutch, and one lexicon for emoticons. Jijkoun and Hofmann (2009) created the Duoman lexicon by bootstrapping scores from the English SentiWordNet (Esuli and Sebastiani 2006) to Cornetto, a Dutch WordNet (Vossen et al. 2008), using automatic translation. This resulted in a lexicon of over 80 000 nouns, verbs and adjectives with real-valued polarity scores between -1 and 1 (e.g. *vergeefs* (EN: *futile*) $\rightarrow -0.02$). To reduce the lexicon size, all words with a polarity in the $]-0.01, 0.01[$ interval were considered neutral and removed, leaving 3 388 entries.

The Pattern library comes packaged with a subjectivity lexicon of Dutch adjectives (De Smedt and Daelemans 2012b). 1 100 adjectives that occur frequently in online product reviews were manually annotated with polarity (-1.0 to 1.0), subjectivity and intensity scores, for each word sense in Cornetto, after which the collection was automatically expanded to 5 500 words. We averaged the scores of adjectives with multiple senses, because no contextual word sense disambiguation would be done. After filtering out words with a polarity in the $]-0.1, 0.1[$ interval, a 2 655-item lexicon remained (containing e.g. *sinister* $\rightarrow -0.5$).

For Bounce, a Twitter sentiment classification system (Kökciyan et al. 2013), a lexicon of emoticons with associated polarity scores (-2, -1, 1 or 2) was manually assembled and made available. We cleaned the list of entries containing rare Unicode characters, and retained 158 emoticons (e.g. `:-))` $\rightarrow +2$).

The polarity information in these three lexicons was used for the following features:

- The ratio of matched positive or negative tokens in a document. For example, if 5 tokens are found in the lexicon, and 3 of those have negative

polarity, the negative ratio is 0.6. We consider tokens, not types, so repeated matches have a higher weight. Ratios are used to normalize for document length and number of matches. Six features are thus obtained: DUO-ratio+, DUO-ratio−, PAT-ratio+, PAT-ratio−, EMO-ratio+ and EMO-ratio−, where DUO and PAT refer to the Duoman and Pattern subjectivity lexicons, and EMO to the Bounce emoticon lexicon.

- For emoticons, we also include raw positive and negative counts (EMO-count+ and EMO-count−).
- The DUO-sum, PAT-sum and EMO-sum features contain the sum of polarity scores of all matched lexicon entries. Unlike the ratios, these sums do not only consider polarity, but also intensity: two matched entries with respective polarities of $+0.2$ and -0.8 would give a negative, not a neutral sum.

The Duoman and Pattern lexicons were matched against the lemmas of the *clean* and *last* text layer, smileys were searched in the *original* layer.

4.3.2 Term extraction

The corpus of transcripts from the CPZ emergency chat hotline (described in Chapter 3) is a resource that contains very relevant words and word sequences, which we wanted to extract as a domain-specific lexicon. To that end, we applied monolingual term extraction on the corpus.

In research on automatic term extraction from text, two methodologies have seen widespread adoption: the linguistic and the statistical approach. The linguistic approach (Daille 1996) starts from a list of language-dependent term formation patterns, which typically consist of part-of-speech patterns (e.g. noun, adjective + noun) in order to identify candidate terms in text. Statistical approaches, on the other hand, extract n-gram sequences as candidate terms and apply statistical measures to filter them. One line of statistical term extraction focuses on *unithood*, i.e. the degree of cohesiveness in multi-word terms, which is determined by applying frequency-based statistics to the single-word components of multi-word terms (Dagan and Church 1994, Pantel and Lin 2001). A second line of research focuses on *termhood*, i.e. the degree in which a given term refers to a domain-specific concept, which is estimated by comparing the frequency of candidate terms in specialized corpora with their distribution in large background corpora containing general vocabulary (Drouin 2003).

We extracted terms using the monolingual component of the TExSIS system (Macken et al. 2013), which is a hybrid terminology extraction system. It

first generates candidate terms from linguistically motivated chunks, and then determines their specificity by combining various statistical filters, including log-likelihood ratio for detecting single-word terms that are distinctive enough (Daille 1996), and C-value (Frantzi and Ananiadou 1999), which handles the extraction of nested terms by examining the frequencies of its components.

TExSIS by default uses the newspaper section of SoNaR (Oostdijk et al. 2013) as a background corpus. We found that many colloquial or conversational words were suggested as terms (e.g. interjections, personal pronouns or verbs in the first and second person), likely because those were rare or absent in the background corpus. This was remedied by using CGN, the Spoken Dutch Corpus (Oostdijk 2000), as a background corpus instead.

The output produced by the TExSIS system contained 8 892 candidate terms. We noticed that POS-tagging accuracy was affected by noise in the corpus. Many misspelled words were tagged as *N(eigen)* (named entities) or *SPEC* (special tokens such as foreign words or part of named entities), and consequently given a high termhood score. Candidate terms that contained these POS tags were therefore removed. The majority of the remaining terms had been POS-tagged correctly.

We further filtered the terms by frequency ($n \geq 3$) and by length: terms should contain at least two content words (i.e. nouns, adjectives, adverbs or verbs). Only multiword terms were considered for inclusion in the domain lexicon, because single words should be handled by the bag-of-words features. All terms were lowercased, lemmatized with Pattern, and reduced to content words only. We thus obtained a lexicon of 251 multiword terms that could be considered very domain-specific. The resulting lexicon can be consulted in Appendix B.

Binary features were made for all terms, indicating occurrence of the term in the content word lemmas of the clean layer. We distinguish three types of term match features, designated with the prefix TERM:

- TERM-exact, for exact matches where all components of the term are found in contiguous order
- TERM-local, for local matches where the constituent components are found in random order within a context of 5 content words, in order to allow e.g. inversion or word insertions.
- TERM-global, a further relaxation that registers a match if all components are found in the entire document.

4.4 Topic model features

The bag-of-words features that were introduced in Section 4.2 have a number of limitations. They result in very sparse feature vectors: a substantial majority of the feature values of an instance will be zero. Additionally, they cannot capture semantic relatedness, such as synonymy. Instances that deal with the same topic are likely to contain semantically related words, but they will not necessarily have any topical words in common. Consider examples 22 and 23, which contain the words *cutting* and *knife*, respectively:

(22) NL: Bang om te leven
Super veel pijn
Bang om iets te zeggen
Jezelf *snijden* en pijn doen
Gewoon omdat jet allemaal niet meer aankan

EN: Afraid to live
Lots of pain
Afraid of saying something
Cutting and hurting yourself
Just because you can't cope anymore

(23) NL: Zonder er bij te kijken pak ikeen grote glimmende *mes*.
Ik voel met mijn vinger hoe scherp het is.
Er komt een mooi drupje rood bloed
Met volle gracht *[sic]* gaat het door mijn hart.

EN: Without looking, I take a big shiny *knife*.
With my finger, I feel how sharp it is.
A nice drop of red blood appears
With full force it stabs my heart.

In bag-of-words representations, only exact matches are taken into account when measuring vector similarity. Semantically related words are therefore interpreted as completely unrelated, and their presence will not increase the similarity score. Statistical *topic models*, models for discovering the abstract topics that occur in a document collection, have been proposed as a solution to these issues.

Latent Semantic Analysis

Latent Semantic Analysis or LSA (Landauer and Dutnais 1997, Landauer et al. 1998) (sometimes also referred to as Latent Semantic Indexing or LSI) starts

from the distributional hypothesis that words that are close in meaning will occur in similar contexts. In order to compare the distributions of words, a term-document matrix is first created where the rows represent unique words (*terms*), columns represent documents, and cells contain the word counts per document. This is, in effect, a bag-of-words vector space with frequency counts. The documents should be drawn from a corpus containing the topics we are interested in modelling. Term frequencies are typically normalized with the tf-idf weighting scheme, as described in Section 4.2.

Next, a condensed representation of the feature space is constructed, by reducing its dimensionality with the *singular value decomposition* (SVD) technique. This makes it possible to infer much deeper (*latent semantic*) relations between features.

SVD transformation is the core component of Latent Semantic Analysis, because it has characteristics that make it attractive for addressing the aforementioned problems with bag-of-words features:

1. Because of sparsity, the predominant feature value is zero. SVD reduces the high dimensionality of the feature vectors by keeping the most relevant information. This way we can both deal with *data redundancy* (similar features will be collapsed in the same dimension) and apply some kind of *smoothing* by removing non-informative features.

2. SVD is capable of capturing *latent* and higher-order associations between terms. Consequently, it allows to find hidden associations (e.g. synonyms) between different instances.

Formally, SVD works by decomposing a given $m \times n$ term-by-document matrix X into the product of three new matrices:

$$X = USV^T \tag{4.2}$$

where U is the $m \times r$ matrix whose columns are orthogonal eigenvectors of XX^T (called the left singular vectors), S is a diagonal $r \times r$ matrix whose diagonal elements are the r singular values of X, that are represented in descending order, and V^T is the transpose of V, the $r \times n$ matrix whose columns are orthogonal eigenvectors of X^TX (called the right singular vectors).

The matrices U and V thus represent terms and documents in a new space, where U contains the terms represented in the latent space (rows of X) and V contains the documents in the latent space (columns of X).

Figure 4.1 illustrates the singular value decomposition of the $m \times n$ matrix X.

The matrix S is the diagonal matrix containing exactly r singular values, where r is the number of linearly independent rows or columns, or *rank* of X.

$$\underset{m \times n}{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix} = \underset{m \times r}{U} \begin{pmatrix} u_{11} & \dots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix} \underset{r \times r}{S} \begin{pmatrix} s_{11} & 0 & \dots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix} \underset{r \times n}{V^{\mathsf{T}}} \begin{pmatrix} v_{11} & \dots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}$$

Figure 4.1: Singular value decomposition of the $m \times n$ matrix X with rank r.

A more intuitive explanation of SVD consists in viewing SVD as a 'process where the axes are rotated in the n-dimensional space. The largest variation among the documents are represented along the first axis, the second largest variation along the second dimension and so forth until the last singular value' (Lopez de Lacalle 2009).

After computing the SVD of a matrix, we can reduce its dimensions by keeping only the first k singular values. Since these are arranged in descending order along the diagonal of S, an order which is retained in U and V^T, keeping the first k singular values is equivalent to keeping the first k rows of S and V^T and the first k columns of U. The most important dimensions that result from the SVD reduction are supposed to represent *latent semantic dimensions* or the most important concepts related to the documents and terms.

More formally, we achieve dimensionality reduction by taking the *rank-k approximation* or *reduced SVD* of X. It is obtained by selecting the first k singular values from S and the first k columns and rows of U and V, thus removing the noisier dimensions:

$$X_k = U_k S_k {V_k}^T \tag{4.3}$$

We thus pass from the original vector space defined by X to the k-dimensional reduced space X_k or the *latent semantic space* of X. By representing the terms and documents in a low-dimensional vector space, words with similar distributional patterns are projected into the same dimension.

Constructing a background corpus for LSA

LSA can be used to extract k latent semantic topics present in a corpus. If these topics are to be useful as features, they should be extracted from a corpus containing topics relevant for suicidality modelling. In absence of a large background corpus containing suicidal material, we used the BootCaT toolkit

(Baroni and Bernardini 2004) to crawl a corpus of web documents about suicide. BootCaT is a suite of perl programs that takes a set of seed terms, combines them into automated queries to the Bing search engine, and retrieves a portion of the found web pages. It then parses the HTML code to extract text, stripping content that is unlikely to be of interest, such as menus, navigation bars, ads, disclaimers and automatic error messages.

As seed terms, we used terms extracted from the chat transcripts corpus using TExSIS (see 4.3.2), which where filtered on frequency ($n \geq 10$) and POS tag (terms should contain a noun), and manually checked, so as to obtain highly specific words or phrases only. This resulted in 105 search terms, thirteen of which where considered unequivocally linked to suicide and added to a whitelist (e.g. *afscheidsbrief, zelfmoordneigingen* and *zelfmoordpoging*; EN: *farewell letter, suicidal tendencies* and *suicide attempt*). BootCaT was configured to find 50 Dutch-language web pages for each query, and retain only those which contained at least one word from the whitelist. We thus collected a background web corpus of 2 603 unique documents, containing over two million words.

Implementation

For the construction of the latent semantic features, we used Gensim (Rehurek and Sojka 2010), a robust and efficient Python package to perform semantic modelling on plain text corpora[1]. The background web corpus extracted with BootCaT was combined with the chat transcripts corpus, tokenized, and represented as a bag-of-words space. Next, a tf-idf model was built on the corpus, to turn the integer-valued frequency vectors into real-valued ones, such that features which were rare in the background corpus would have their value increased. Finally, the tf-idf-weighted bag-of-words space was transformed into a latent space of lower dimensionality with LSA. We experimented with four ranks: 20, 50, 100 and 200, resulting in as many latent semantic topics.

The text in the clean layer of each instance was mapped to the bag-of-words space, weighted using the tf-idf model, and projected into each of the four LSA spaces. Formally, we projected each document vector $\overrightarrow{d}$ into the reduced space of rank k by applying the following transformation:

$$\overrightarrow{d}_k = {S_k}^{-1} {U_k}^T \overrightarrow{d} \tag{4.4}$$

As a result, the sparse feature vector $\overrightarrow{d}$ turns into a dense feature vector $\overrightarrow{d}_k$ in the low-dimensional space. A value at position n in the vector indicates the relevance of the document to topic n. We can now determine the similarity

[1] http://radimrehurek.com/gensim/

between pairs of vectors (documents), or the similarity between a specific vector and a set of other documents. We derive two types of features:

- The k topic scores in the vector (LSA-20, LSA-50, LSA-100 and LSA-200).
- The average similarity between a document and the 290 documents in the chat transcripts corpus (LSA-20-avg, LSA-50-avg, LSA-100-avg and LSA-200-avg)

These should allow a classifier to learn which latent semantic topics (rather than e.g. words) are salient for a task, and to what extent the document aligns with the topics in the CPZ emergency chat conversations.

4.5 Other features

We hypothesized that the presence of names of locations, organizations and well-known persons could help in recognizing journalistic and informative texts, as well as personal texts about celebrities. All documents were therefore processed with DBpedia Spotlight, a tool to link concepts in a text to the DBpedia ontology (Mendes et al. 2011). DBpedia is a hub in the Web of Data that contains structured encyclopedic knowledge from Wikipedia. We used the tool to only link to the *person*, *organization* and *populated place* classes in the ontology. In the output, a number of supposed named entities with a high frequency were in fact common words, misspellings or interjections, typical for noisy online language (e.g. *Derna, Ela, Grust, Haa, Ist, Jaah, Kem, Menen, Nice, Slape, VN, Well, Zalk* and *Zenne*). These were removed. We defined three features based on the ontology linking: one binary feature indicating the presence of one or more named entities in the post (NE-presence), and two integer features for the number of found named entities: NE-count for all matches (including repeated entities), and NE-unique for unique matches.

Apart from the lexical and semantic features described above, three features were added to describe basic surface properties of the original text. Post length, defined as the logarithm of the number of characters (LENGTH), could serve to avoid incorrectly classifying very short posts as suicide-related. The ratio of capitalized characters (CAPS-char) and of tokens with more than one capitalized letter (CAPS-token) describe the use of regular, absent or excessive capitalization.

Feature group	layer	type	size	Feature group	layer	type	size
W1	clean	binary	51 670	EMO-ratio+	orig.	real	1
W2	clean	binary	300 277	EMO-ratio−	orig.	real	1
W3	clean	binary	475 167	EMO-sum	orig.	real	1
LEM1	clean	binary	50 739	EMO-count+	orig.	integer	1
LEM2	clean	binary	289 438	EMO-count−	orig.	integer	1
LEM3	clean	binary	481 639	TERM-exact	clean	binary	131
WCH2	orig.	binary	3 383	TERM-local	clean	binary	159
WCH3	orig.	binary	32 576	TERM-global	clean	binary	200
WCH4	orig.	binary	136 426	LSA-20	clean	real	20
LCH2	clean	binary	1 799	LSA-50	clean	real	50
LCH3	clean	binary	18 136	LSA-100	clean	real	100
LCH4	clean	binary	92 049	LSA-200	clean	real	200
PAT-ratio+	clean	real	1	LSA-20-avg	clean	real	1
PAT-ratio−	clean	real	1	LSA-50-avg	clean	real	1
PAT-sum	clean	real	1	LSA-100-avg	clean	real	1
DUO-ratio+	clean	real	1	LSA-200-avg	clean	real	1
DUO-ratio−	clean	real	1	NE-presence	clean	binary	1
DUO-sum	clean	real	1	NE-count	clean	integer	1
PAT-ratio+(last)	last	real	1	NE-unique	clean	integer	1
PAT-ratio−(last)	last	real	1	LENGTH	orig.	real	1
PAT-sum (last)	last	real	1	CAPS-char	orig.	real	1
DUO-ratio+(last)	last	real	1	CAPS-token	orig.	real	1
DUO-ratio−(last)	last	real	1				
DUO-sum (last)	last	real	1	Total ($n = 46$)			1 934 186

Table 4.2: Overview of the feature groups. Features are based on the *original*, *clean* or *last* text layer, and can be integers, real-valued or binary.

4.6 Feature counts

In summary, we defined 46 feature groups. Table 4.2 gives an overview of the groups, the text layer they are based on, the number of features in the group, and the feature value type.

Full feature vectors consist of 1 934 186 individual features, the bulk of which ($> 99.9\%$) are part of the binary bag-of-words feature groups. In Section 6.2, we discuss feature selection, which is aimed at removing irrelevant features, reducing feature vector size and improving classification performance.

4.7 Summary

In this chapter, we described the selection of information sources to derive features from. The aim was to obtain features that were salient (i.e. informative to distinguish posts belonging to different classes) and to provide abstraction from the training data (to improve generalization and reduce data sparsity).

After removing noise, three preprocessing steps were applied to the data: tokenization, part-of-speech tagging and lemmatization. Features could then be calculated on four text layers: the original, cleaned and lemmatized full text, and the last 10 tokens of cleaned text, which may contain a summary of the overall topic or mood of a post.

The following categories of features were defined: (1) token and character ngram bag-of-words features of various length, providing low-level information without abstraction, (2) lexicon-based features to obtain an abstract representation of the polarity in a post, and to flag relevant suicide-related terms, (3) topic model features that group semantically related concepts together, and (4) features that describe the length, capitalization and presence of named entities in a post.

Having defined the feature vectors for our experiments, we continue with a description of the machine learning techniques in the next chapter.

CHAPTER 5

Machine learning techniques

The principle of classification-based supervised machine learning was introduced in Chapter 4: in the training phase, a learning algorithm is fed training instances, represented as vectors of disambiguating features and the correct class label. During classification, the algorithm predicts the labels of unseen test instances, informed by the training information.

All supervised machine learning methods aim to find the optimal function to map from the multidimensional feature space to the unidimensional class label, known as the *hypothesis* or *model*. Optimal generally refers to a good balance between *bias* and *variance*, which tend to be negatively correlated. A model should be able to accurately fit the regularities in the training set (i.e. low bias: no underfitting and low training error rate), but should also generalize well to unseen documents (i.e. low variance: no overfitting to noise or high model complexity). Learning methods vary considerably in the way they induce a hypothesis from the training data. Most methods adhere to the principle of *minimum description length*, which drives them to use the set of hypotheses for which the description of the chosen function together with the number of training errors is shortest. This is known as *eager* learning. *Memory-based learning*, on the other hand, stores all training instances in memory, without abstracting or eliminating noise and exceptions, and is therefore known as *lazy*

learning.

In the text classification literature, linear classifiers are the most commonly used method. Representative examples include support vector machines (SVMs) with linear kernels, naive Bayes methods and boosted linear classifiers. Non-linear classifiers, such as SVMs with non-linear kernels, decision trees, neural networks and memory-based learning have also been successfully applied to the task. The k-nearest neighbour (kNN) method, a type of memory-based learning, is particularly popular due to its intuitive simplicity and competitive prediction accuracy. However, properly tuned linear classifiers have been observed to achieve similar to better performance compared to non-linear ones, and are typically sufficient for solving practical text categorization problems. They have the added advantage of being computationally efficient, during training as well as testing (Joachims 1998, Yang and Liu 1999, Schapire and Singer 2000, Zhang and Oles 2001, Sebastiani 2002, Yang and Joachims 2008).

The *no free lunch theorem* (Wolpert and Macready 1995) postulates that no single inductive algorithm is universally better than any other. It is therefore necessary to experimentally determine the best algorithm for a given classification task. Yang and Liu (1999) performed a controlled study on five common text categorization methods (SVM, kNN, neural networks, linear least-squares fit (LLSF) and naive Bayes), and evaluated their robustness in dealing with skewed category distributions and their performance as a function of the training set category frequency. They found that SVM, kNN and LLSF significantly outperformed neural networks and naive Bayes classifiers, when the number of positive training instances was small. In earlier classification experiments on our dataset, we experimented with SVM, kNN and naive Bayes classifiers. The results corroborated that naive Bayes performance consistently trailed that of kNN and SVM (Desmet and Hoste 2014). We therefore chose to focus our experimental investigation on support vector machines (with linear and polynomial kernels) and memory-based learning.

In this chapter, we provide a theoretical description of the two machine learning algorithms that were used (Sections 5.1 and 5.2), their implementations and the hyperparameters that were tested. We describe the techniques to measure and validate classification performance in Section 5.3. Cascaded classification, a classifier ensemble method, is discussed in Section 5.4.

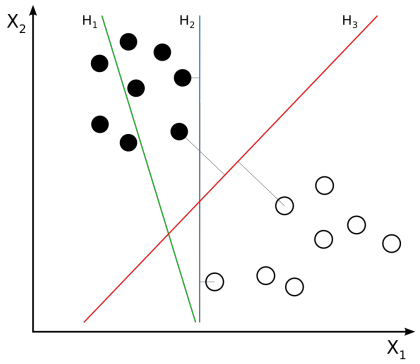

Figure 5.1: Possible classification hypotheses as decision boundaries in a two-dimensional feature vector space.

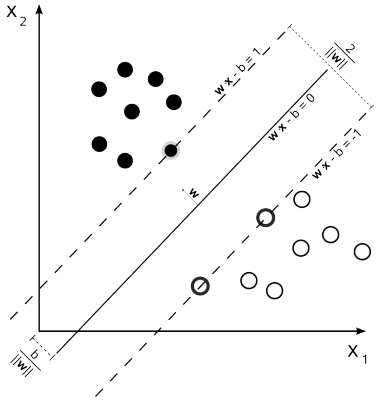

Figure 5.2: Support vectors and maximized margin.

5.1 Support vector machines

5.1.1 Theory

Support vector machines are a supervised learning approach for binary classification problems. It was introduced by Vapnik (1995), and is based on the *structural risk minimization* principle, which aims to simultaneously minimize the VC-dimension of a classifier (a measure for how complicated it can be) and its error rate on the training data.

In the vector space defined by the training instances, SVM aims to find the best decision surface to separate the instances into both classes. The input vector space is n-dimensional, where n is the number of features, so the decision boundary could be an $n-1$-dimensional hyperplane that linearly separates the space into two half spaces. At the time of classification, new data points are mapped in the vector space and their class is determined based on which side of the hyperplane they fall on. The distance between the data point and the hyperplane can be used as a measure for classification certainty.

Consider for example the vector space in Figure 5.1, which for simplicity is two-dimensional. The instances are represented with two features (displayed along axes X_1 and X_2), and the separating hyperplane must be one-dimensional, i.e. a line. In this case, the instances are linearly separable, but the rationale can be generalized to high-dimensional spaces where the data points need not be linearly separable.

The first hypothesis H_1 does not separate the positive from the negative instances. H_2 and H_3, on the other hand, are successful decision boundaries, although they have a different *margin*. The margin is the amount a hyperplane can move in either direction without causing it to misclassify some of the data (as illustrated in Figure 5.2). SVM tries to find the hypothesis that maximizes the margin, which in this case would be H_3. Interestingly, linear SVM only uses the training instances that are on the boundaries of the margin. These are called the support vectors (marked with a thicker circumference in Figure 5.2), and all other data points in training essentially have no effect on the decision boundary formulation. This is in stark contrast to memory-based learning, where every training instance influences the hypothesis (cf. infra).

Formally, the decision boundary to be produced by SVM can be written as follows:

$$\overrightarrow{w} \cdot \overrightarrow{x} - b = 0 \tag{5.1}$$

where $\overrightarrow{x}$ is the feature vector of an instance which must be classified, and the weight vector $\overrightarrow{w}$ and constant b are inferred from the training data $D = (y_i, \overrightarrow{x}_i)$, containing pairs of training feature vectors $\overrightarrow{x}_i$ with associated class labels y_i. Class labels y_i are either $+1$ for positive instances, or -1 for negative instances. The optimization problem can then be defined as finding a $\overrightarrow{w}$ and b such that

$$\overrightarrow{w} \cdot \overrightarrow{x} - b \geq +1 \quad if \quad y_i = +1 \tag{5.2}$$
$$\overrightarrow{w} \cdot \overrightarrow{x} - b \leq -1 \quad if \quad y_i = -1 \tag{5.3}$$

and that minimizes the vector 2-norm of $\overrightarrow{w}$.

This can be solved for linearly separable datasets (Joachims 1998). In order to accommodate non-linearly separable problems, two extensions to SVM have been proposed. The first method is *soft margin hyperplanes*, which relaxes the above constraints and allows classification errors during training, in order to construct a more efficient hyperplane. The soft margin variant thus permits a trade-off between error rate on the training data and margin maximization, which is controlled by a cost parameter C.

The second method is known as the *kernel trick*: an arbitrary kernel function is used to replace the original feature vector representation with a higher-dimensional space that allows the modelling of interactions between features, so that the data becomes linearly separable. A linear kernel function would be generalized as follows:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \tag{5.4}$$

where x_i and x_j are two training vectors that are combined, and $\phi(x)$ is the transformation function to map x into a higher-dimensional space. Kernel functions can also be non-linear, which may allow better separation.

This means that we can build non-linear SVMs in two steps: first, the data is mapped to a new feature space with the kernel function, and then a regular linear machine is trained in that new space to classify the instances. When used with a kernel function, SVM estimates weights for the training instances in the new feature space, rather than learning a fixed set of weights for all the features in the input. Kernel functions can therefore be seen as a similarity function to compare unseen instances to the training data. Kernel functions can be implemented independently from the classification algorithm, and can be chosen to match a specific classification problem. Aside from the linear kernel, typical choices are the *Gaussian radial basis function* (RBF), *polynomial* and *sigmoid* kernels. For an in-depth discussion of kernel methods, we refer to Cristianini and Shawe-Taylor (2000).

5.1.2 Implementation

In our experiments, we use LIBSVM[1], version 3.17, a popular and extensive SVM software library by Chang and Lin (2011) that has been under active development since 2000. Among other things, it implements one-class, binary and multiclass support vector classification.

Binary SVM classification is used with three of the four supported kernel types: linear, polynomial and sigmoid. We omitted the RBF kernel, although it is a popular choice, because it can be configured to behave like the linear and sigmoid kernels with specific parameters. There are also fewer parameters to be set than for polynomial kernels, which reduces the complexity of model selection. However, there are situations where the RBF kernel is not suitable, particularly when the number of features is very large, as is the case in our study. Hsu et al. (2010) advise to then use linear kernels instead. In Goldberg and Elhadad (2008), it is also stated that in natural language processing research, polynomial kernels are generally more popular than RBF, compared to other fields. We therefore expect the best performance with linear or polynomial kernels, but include the sigmoid kernel for completeness.

[1] Available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

Both non-linear kernels need to be parameterized further, as per the definitions for polynomial (5.5) and sigmoid (5.6) kernel functions,

$$\left(\gamma {x_i}^T x_j + C\right)^d \tag{5.5}$$

$$tanh\left(\gamma {x_i}^T x_j + C\right) \tag{5.6}$$

in which

- γ is a free parameter, which we vary between 2^{-14} and 2^4, stepping by factor 4.
- d is the degree of the polynomial. We vary d between 2 and 5, and expect the best results for lower degrees, since larger degrees tend to overfit on NLP problems (Goldberg and Elhadad 2008).
- c is a constant trading off the influence of higher-order versus lower-order terms. We fix c to the default of 0.

The general (not kernel-specific) configuration for the SVM algorithm is as follows:

- We use the soft margin method to allow training errors when constructing the decision boundary, and vary the associated cost parameter C between 2^{-6} and 2^{12}, stepping by factor 4. This is the only parameter we vary when using linear kernels.
- Shrinking heuristics are always used. Shrinking is a technique to reduce the training time: by identifying and removing some bounded elements in the optimization problem, it becomes smaller and can be solved in less time.
- The stopping criterion ϵ is set to 0.001. Because the optimization method only asymptotically approaches an optimum, it is terminated after satisfying this stopping condition.

All data sets were scaled before applying SVM, i.e. all feature values were linearly mapped to the range $[0, 1]$. All binary features already fit this range, but the remainder of the features had varying minimum and maximum values. The LSA features in particular presented varying spreads. With unscaled features, values in greater numeric ranges dominate those in smaller numeric ranges. Another advantage of scaling is that it prevents numerical problems during the SVM calculation.

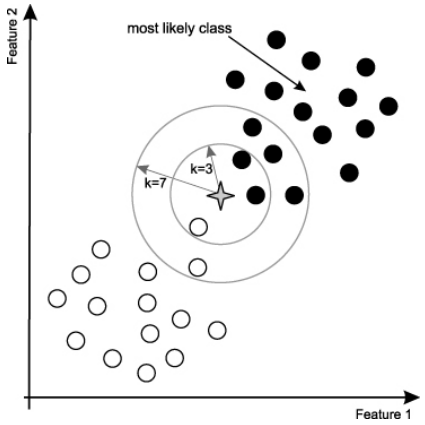

Figure 5.3: Examples of k Nearest Neighbour binary classification in a two-dimensional feature vector space. The star represents an unseen instance to be classified.

5.2 Memory-based Learning

5.2.1 Theory

Memory-based learning (MBL), also known as instance-based or exemplar-based learning, is a machine learning paradigm that is commonly applied with the *k Nearest Neighbour* (kNN) method. The intuition behind the algorithm is straightforward: first, all training instances are stored in memory. A test instance is classified by searching the memory store for the most similar training instances (i.e. the k nearest neighbours) and assigning one of the classes associated with the neighbours. When $k = 1$, the class of the single nearest neighbour is assigned, whereas with larger values of k, each neighbour casts a vote for its own class label. Figure 5.3 provides a simple illustration for binary classification between black and white: when $k = 3$, the test instance will be assigned to the black class, since it is the class two out of the three nearest neighbours belong to.

As opposed to SVM, memory-based learning does not form a generalized hypothesis, as it performs no abstraction when storing the training data. Rather, it constructs hypotheses directly from the training instances, which allows the hypothesis complexity to grow with training set size. Because induction is delayed to the classification phase, the method is often referred to as *lazy learning*. As a consequence, runtime during classification grows with the number of exemplars, making it relatively slow to use. MBL has the advantage that it functions

similarly to how humans learn from previous experience. This makes its classification decisions intuitively understandable, unlike classifiers that may operate more like a black box. Daelemans et al. (1999) show that the lack of generalization in memory-based learning also makes it eminently suitable for natural language classification tasks, because the exceptions that are ubiquitous in many language problems are not forgotten.

The operationalization of kNN depends on three key components:

- the value for k
- the *distance metric* that defines how similarity between vectors is measured, in order to determine which training instances are nearest to the test instance
- the voting strategy that specifies how a class label is extrapolated from the nearest neighbours, when $k > 1$.

Number of nearest neighbours

The number of nearest neighbours k that is considered for prediction is typically set to 1, i.e. the class of the most similar training instance is assigned. Larger values for k (usually an odd number, to avoid ties) reduce the impact of noisy training instances, and make the decision boundary smoother. In other words, the complexity of a kNN model is highest with $k = 1$, and decreases as k goes up. With $k = 1$, a model's training error rate is 0 (no bias), but it may be overfitting the data. Increasing k will likely introduce training errors, but the resulting model may be more robust. Hoste et al. (2002) have shown that no single value of k works best for all data sets. The optimal k-value is therefore best determined experimentally for a given data set.

Distance metric

Given a test instance x_t, the distance (or similarity) to every training instance x_j needs to be calculated in order to find the nearest neighbours. The distance between two vectors is measured with a distance metric.

The most basic metric is *overlap*. For symbolic features, it defines the distance between x_t and x_j as the number of non-identical features. In other words, it takes the sum of all mismatched feature values, and considers every difference in feature value to be equally important. This all-or-nothing approach would not work well with numeric features, because it discards the similarity information

that is inherent to scale (integers 1 and 2 would be considered as dissimilar as 1 and 10), and with real-valued numbers, the probability of an exact match is minimal. Therefore, overlap for numeric features is calculated as per Equation 5.7, i.e. the sum of normalized differences per feature.

$$\Delta(x_t, x_j) = \sum_{i=1}^{n} w_i abs\left(\frac{x_{t_i} - x_{j_i}}{max_i - min_i}\right) \tag{5.7}$$

The term w_i refers to the weight associated with feature i. Without feature weighting, a metric considers each feature to be equally important. However, it is usually the case that some features are more informative for the classification than others. Two strategies to address this are feature selection, where less informative features are removed, and feature weighting, which gives important features a greater impact on the similarity calculation. Both strategies are discussed in detail in Chapter 6.

The *dot product* and *cosine* distance metrics are two other metrics that were developed specifically for numeric features (Daelemans et al. 2009). The dot product (or inner product) of vectors x_t and x_j will be higher with better matches, so it needs to be inverted to describe distance. This is achieved by subtracting it from the maximum attainable dot product, namely that of an exact match. Equation 5.8 formally defines the dot product distance metric.

$$\Delta(x_t, x_j) = dot_{max} - \sum_{i=1}^{n} w_i x_{t_i} x_{j_i} \tag{5.8}$$

A property of the dot product metric is that when either x_{t_i} or x_{j_i} is zero, the feature does not contribute to the dot product. This differs significantly from the overlap metric, because non-matching values where one has a zero value are ignored here, whereas they count as much as any other value mismatch in the overlap metric. This makes the dot product metric better suited for sparse vectors (e.g. our binary bag-of-words features), since the large proportion of zero-valued features cannot dominate the distance.

The cosine metric is a variant of the dot product metric that corrects for large differences in the length of the instance vectors. It divides the dot product metric by the product of the length of the two vectors. This similarity value is again converted to a distance by subtracting it from a cos_{max} term, as in Equation 5.9.

$$\Delta(x_t, x_j) = cos_{max} - \frac{\sum_{i=1}^{n} w_i x_{t_i} x_{j_i}}{\sqrt{\sum_{i=1}^{n} w_i {x_{t_i}}^2} \sqrt{\sum_{i=1}^{n} w_i {x_{j_i}}^2}} \tag{5.9}$$

Voting strategy

When $k > 1$, a strategy is needed to pick one class label from the set of labels suggested by the k nearest neighbours. The simplest approach is normal majority voting, which means that all neighbours have equal weight and the most frequent class label is chosen. Ties are broken by assigning the label that has the highest frequency in the entire training set.

Performance usually benefits from larger values of k, because it provides smoother local estimates. However, the local neighbourhood may become large in sparsely populated regions, causing the majority voting method to produce overly smooth results. A more informed voting strategy is *distance weighted voting*, which gives the closer neighbours a stronger vote than distant neighbours, and thus reduces the sensitivity of a learner to the k parameter.

Dudani (1976) proposed two strategies. With *inverse linear* weighted voting, the closest neighbour gets a weight of 1, the furthest a weight of 0, and the weights of the neighbours in between are scaled linearly to that interval, according to their distance from the query instance. With *inverse distance* weighted voting, the weight of a neighbour is defined as $w_j = 1/(d_j + \epsilon)$, where d_j is the distance from the test instance and ϵ is a small constant to prevent division by zero.

5.2.2 Implementation

TiMBL[2] is a software package for efficient kNN classification (Daelemans et al. 2009). It implements kNN as per the IB1 algorithm described by Aha et al. (1991), with the difference that it considers all neighbours at the k nearest distances, rather than restricting the amount of neighbours to an absolute number k, potentially disregarding instances that are tied at the same distance. TiMBL can be extensively configured with the hyperparameters described in Section 5.2.1. It also provides various algorithmic optimizations for the nearest neighbour search, and alternative ways to structure the instance memory to speed up classification. IGTREE compresses the instance base into a decision tree (with features ordered on information gain), and TRIBL and TRIBL2 are hybrids between IB1 and IGTREE. The compression usually comes at a small cost in terms of accuracy, but is nevertheless attractive given the considerable speedup it produces, especially on large datasets. Unfortunately, IGTREE and its derivatives are not suited for numeric features, because numbers are simply treated as literal strings (Daelemans et al. 2009). We therefore limit ourselves to experimenting with IB1.

[2] Available from http://ilk.uvt.nl/timbl/

We use TiMBL version 6.4.3, and allow the following IB1 hyperparameters:

- Distance metrics (m): overlap (O), dot product (D) and cosine (C)
- Feature weighting (w): no weighting ($None$), gain ratio (GR), information gain (IG) and chi-squared (χ^2) (for a detailed discussion of these metrics, we refer to Section 6.2.1)
- k-values: 1, 3, 5, 7 and 9
- Voting strategies (d): normal majority (Z), inverse distance (ID) and inverse linear (IL)

5.3 Model validation

In order to measure the performance of a classifier, we can use evaluation metrics that compare its predicted results to the gold standard solutions. These results are typically represented in an *error matrix* or *confusion matrix*, in which each column represents the instances in a predicted class, while each row represents the instances of the actual gold standard class. Table 5.1 provides an example matrix for a binary classification task ($True$ or $False$), e.g. for the detection of alarming suicidality in a post.

		prediction	
		True	False
gold standard	True	10	3
	False	7	80

Table 5.1: Confusion matrix for a binary classification task.

Along the diagonal of the matrix, we find the amount of instances that have been classified correctly: 10 true positives and 80 true negatives. We can also distinguish two kinds of errors. False positives ($n = 7$), also known as type I errors or false alarms, occur when the classifier fails to reject a true null hypothesis, i.e. it indicates that the post contains suicidality while there is none. False negatives or type II errors ($n = 3$), on the other hand, occur when the classifier fails to detect suicidality that is present. In an application of our task, where posts are filtered for review by a suicide prevention worker, false negatives can be considered more problematic than false positives, since the latter category can still be ignored.

The choice of evaluation metric should be motivated by the task it is used for. Many natural language processing problems (e.g. part-of-speech tagging) are evaluated in terms of accuracy, which is defined in Equation 5.10 and applied in 5.11.

$$accuracy = \frac{true\ positives\ +\ true\ negatives}{total\ number\ of\ instances} \tag{5.10}$$

$$accuracy = \frac{10 + 80}{10 + 3 + 7 + 80} = \frac{90}{100} = 0.9 \tag{5.11}$$

Accuracy values correctly classified negatives as much as true positives. For a skewed dataset such as ours, where the number of positive versus negative instances is highly unbalanced, this is not a desirable property, since the number of true negatives will dominate the accuracy and error of the classifier. The accuracy of a baseline classifier that always predicts the majority class, for example, will increase with skewness. In our running example, a majority class baseline would achieve an accuracy of 0.87 without detecting any alarming posts. For this reason, the recall, precision and F-score measures are more commonly used in text categorization evaluations (Van Rijsbergen 1979).

We are interested in the assignment of posts to the positive category. Precision measures the ratio of correct assignments by the classifier, divided by the total number of the classifier's assignments (Equations 5.12 and 5.13). It is a measure for the amount of irrelevant hits that a system produces, i.e. how noisy it is. Recall, also called sensitivity, is defined to be the ratio of correct assignments by the classifier divided by the total number of gold standard assignments (Equations 5.14 and 5.15).

$$precision = \frac{true\ positives}{true\ positives\ +\ false\ positives} \tag{5.12}$$

$$precision = \frac{10}{10 + 7} = \frac{10}{17} \approx 0.59 \tag{5.13}$$

$$recall = \frac{true\ positives}{true\ positives\ +\ false\ negatives} \tag{5.14}$$

$$recall = \frac{10}{10 + 3} = \frac{10}{13} \approx 0.77 \tag{5.15}$$

F-score is a measure that combines precision and recall, as per Equation 5.16. The β term determines the weight of precision versus recall. Traditionally, F-score is calculated with $\beta = 1$, resulting in a harmonic mean of precision and recall, known as F_1 score or balanced F-score (Equations 5.17 and 5.18).

We report F_1 score for most of our experiments, since it is the standard and widespread F-score implementation. Different values for β, with the corresponding increased focus on precision or recall, would have to be motivated by the preference of an end user reviewing messages flagged by the system: one can prefer speed (high precision) over thoroughness (high recall), or vice versa.

For some evaluations, we use the F_2 measure to give more weight to recall (Equations 5.19 and 5.20). We do this for the cascaded experiments, which are described in detail in Section 5.4. In cascades, false negatives (affecting recall) can be expected to be more detrimental to overall performance than false positives, so optimizing for better recall is a logical strategy.

$$F_\beta = \left(1 + \beta^2\right) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision \ + \ recall} \tag{5.16}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision \ + \ recall} \tag{5.17}$$

$$F_1 = 2 \cdot \frac{0.59 \cdot 0.77}{0.59 + 0.77} \approx 0.67 \tag{5.18}$$

$$F_2 = 5 \cdot \frac{precision \cdot recall}{4 \cdot precision \ + \ recall} \tag{5.19}$$

$$F_2 = 5 \cdot \frac{0.59 \cdot 0.77}{4 \cdot 0.59 + 0.77} \approx 0.73 \tag{5.20}$$

An important consideration when assessing the performance of a machine learner is whether the validation is carried out on a representative test sample. The simplest validation method is data splitting. It consists in splitting the original data into three distinct samples: a training set, on which the model is developed, a development test set for parameter tuning and model selection, and a separate test set for model validation, to which the model has not been specifically tuned. The main problem with this approach is that in order to retain sufficient data for training, the test set may not be large enough to be representative. For binary classification problems, Harrell (2001) suggests that there should be a bare minimum of 100 instances of the minority class in the test sample, preferably many more. Another disadvantage is that the choice of split may have a significant impact on the estimated predictive power of a model, i.e. a different split would produce different assessments.

Cross-validation provides a solution to these problems. It is a generalization of data splitting where p instances are omitted from the data, and their classes

are predicted with a model trained on the remaining $n - p$ instances. The process is repeated until all n instances have been tested. This approach allows all the data to be used both for training and testing. With $p = 1$, this is known as *leave-one-out cross-validation*, which requires n repetitions. A less computationally intensive method is *grouped* or *k-fold cross-validation*, where the data is randomly split into k subsets of roughly equal size, called *folds*. A model is trained on $k-1$ folds and tested on the remaining fold, and the process is rotated k times (Weiss and Kulikowski 1991).

Typically, the overall score obtained after cross-validation is the average of the performance scores calculated on each fold. When a dataset is highly skewed, the distribution of positive instances may be unequal over the folds. This will introduce variance in the overall results, because some models may have access to relatively few positive instances in training, or the test fold may not contain any positive instances, resulting in a zero score for that fold with some metrics (e.g. F-score on the positive class). The former effect can be mitigated by selecting a sufficiently large number of k, so that the relative size of the training set is also large. In natural language processing research, 10-fold cross-validation is the de facto standard (Jurafsky and Martin 2009). To prevent the effects of variable amounts of positive instances in the test fold, stratified cross-validation may be used (such that each fold contains the same ratio of positive versus negative instances), or the performance metric can be calculated on the full data set, obtained by concatenating the results from all folds.

A drawback of cross-validation is that the test set is not entirely blind. In order to build a system, it is customary to examine the training data so as to make informed design choices, e.g. with relation to feature engineering. For this reason, it is still worthwhile to reserve some of the data as a completely independent test set.

In the experimental section of this thesis, performance is measured on the concatenated results after 10-fold cross-validation, and error analysis and scaling experiments are performed on a held-out test set, described in Section 7.1.

5.4 Direct versus cascaded classification

In Chapter 3, we introduced the annotation scheme (3.2.1) and definitions (3.2.3) for the two problems under study: the relevance task, concerned with detecting posts that contain references to suicide in the strict sense, and the severity task, in which posts annotated with a high or intermediate severity risk are to be detected. The annotation scheme has the structure of a multi-level decision tree, where the annotation choice on one level determines whether annotation

ends, or continues on the next level.

Both tasks can be approached as simple classification problems, where a single classifier needs to determine whether a post pertains to the category of interest or not. We will henceforth refer to this approach as *direct classification*. Given the cascaded nature of the annotation scheme, however, it can be worthwhile to combine multiple classifiers in a sequence. In such a setup, classifiers can be trained to predict the annotations at intermediate levels in the decision tree, and influence the final classification accordingly. For the severity task, for example, a classifier for relevance can act as a filter, since all severe posts must also be relevant posts, as per the annotation scheme. This is known as *cascaded classification*.

Cascaded classification is an example of *ensemble learning*, a family of machine learning techniques which start from the hypothesis that a combination of classifiers can obtain better predictive performance than any of its constituent classifiers (Opitz and Maclin 1999). Most ensemble methods (e.g. bagging, boosting, voting or stacking) can be considered multi-expert systems, where multiple classifiers provide a hypothesis for the same classification problem. The assumption is that combining a diverse set of classifiers improves the generalization accuracy, provided that the ensembles members have sufficient individual performance and the errors they make are, to some extent, non-overlapping. The cascade method differs in that it is a multistage system, where the number of instances is reduced in consecutive steps. It has been successfully applied to a number of NLP problems, such as fine-grained named entity recognition (Dinarelli and Rosset 2011, Desmet and Hoste 2013b), the detection of hedging (Tang et al. 2010) and semantic head recognition (Michelbacher et al. 2011).

An advantage of cascaded classification is that the constituent classifiers can be trained to focus on a subtask of the final decision. When a relevance and a severity classifier are cascaded, for example, the latter will receive mostly relevant posts, and can be better tuned to the difference between severe and non-severe posts about suicide. Intuitively, this should result in a hypothesis that is simpler and more robust. It is important to note, however, that any false negatives produced by an upstream classifier cannot be corrected further down the cascade. It would therefore seem advisable to have upstream classifiers that are not conservative in their predictions. Rather, they should aim for high recall at the expense of precision.

An in-depth review of cascaded classifiers is presented in Minguillón Alfonso (2002). It compares the performance of cascaded classifiers to that of decision tree classifiers that operate on the entire problem. The study finds that cascading classifiers improve accuracy, because they decrease both bias (more complex boundaries can be constructed) and variance (the subproblems can be modeled

more robustly). However, cascades typically come at an increased computational cost. Short cascades, consisting of two or three components, are found to be sufficient to improve classification accuracy.

We want to investigate the effect of cascaded classification on performance for the severity task. We hypothesize that a cascaded severity classifier can achieve better performance than a direct classifier. We further want to determine the properties of a good upstream classifier by experimentally verifying the impact of an increased emphasis on recall over precision.

5.5 Summary

This chapter presented the methodology to obtain and validate classification models. We selected two machine learning frameworks for the experiments, support vector machines and memory-based learning, and described the available hyperparameters for these algorithms, which may have considerable impact on performance. Cascades were proposed as an alternative to direct classification, with the potential advantage that models in a cascade derive simpler and more robust hypotheses. Model validation will be done with tenfold cross-validation, and performance measured with the $F_{\beta=1}$ and $F_{\beta=2}$ metrics.

The next chapter discusses the problem of finding the best combination of features and hyperparameters for a classification model.

CHAPTER 6

Learner optimization with genetic algorithms

In the previous chapters, we introduced the building blocks for a system to detect suicidality in social media messages. We collected and annotated a data set, defined the relevance and severity text categorization tasks, and constructed a range of features that we believe can be informative in distinguishing posts between the categories. We also selected two machine learning algorithms to experiment with, a method for combining classifiers in a cascade, and a procedure for model validation. In this chapter, we describe the experimental challenges involved in combining those components into a successful model.

More in particular, we discuss our approach for finding a good combination of algorithm settings and relevant features. These problems are known as *hyperparameter optimization* and *feature selection*, respectively, and the potential of both techniques to improve classification performance has been amply demonstrated in previous machine learning research. We refer to the work of Hoste (2005) for a detailed discussion of their methodological importance in experiments on machine learning of natural language.

In Sections 6.1 and 6.2, we study hyperparameter optimization and feature selection. Section 6.3 demonstrates that the amount of variables to be optimized constitutes a non-trivial search problem. We discuss a number of possible so-

lutions, and in Section 6.4 elaborate on genetic algorithms, the evolutionary approach to search we adopted in our work.

6.1 Hyperparameter optimization

As mentioned in Chapter 5, supervised machine learning algorithms build a model by finding the optimal function to map a vector of features values to the correct class. A typical learner therefore needs to optimize the parameters of this function, so as to minimize the training error. Hyperparameters, on the other hand, are the settings of the algorithm itself. Some have an influence on practical aspects of running the algorithm, such as speed or required memory, but more importantly, they can affect performance. Hyperparameters like the cost value C for SVM or the number of nearest neighbours k for MBL, for example, influence the capacity of a learner to fit the training data, and can be tuned with the goal of preventing underfitting (the model does not capture underlying trends in the training data) and overfitting (the model is overly complex and fits noise in the data), so as to achieve good generalization. For the task of detecting high-risk suicidal content, for example, a model suffering from overfitting would only be capable of detecting posts with features (e.g. words) that are very similar to the ones found in specific positive training instances, and suffer from low recall as a result.

Changing hyperparameters, therefore, can have a dramatic effect on classifier performance. Although most machine learning implementations are configured to use sensible hyperparameters by default, these settings are not guaranteed to be optimal for a particular problem. In effect, it is hard to predict how hyperparameters will perform, and rules of thumb are rare. It is therefore recommended to test them, by doing some kind of hyperparameter search. This is known as hyperparameter optimization or *model selection*.

Hoste et al. (2002) argue that it is unwarranted to compare machine learning algorithms (e.g. SVM and MBL) without doing optimization first. A difference in performance between classifiers of distinct algorithms cannot readily be explained by the proposition that one algorithm must be better suited to the task. In other words, such simple comparisons may lead to flawed conclusions about a paradigm's innate ability to model the problem, i.e. it having the right 'bias'. Rather, it may be the case that performance differences between configurations of the same algorithm are equally or more significant. When the goal is to find the optimal algorithm for a given task, it is therefore good practice to compare them after hyperparameter optimization.

In Sections 5.1.2 and 5.2.2, we described the hyperparameters for LIBSVM and

TiMBL that will be tested in our experiments. To recapitulate, for SVM, we allow 3 kernels, 10 cost values C, 10 γ values and 4 degrees of freedom d. Considering the compatibility of the kernels with the other hyperparameters, the following amounts of combinations are possible: 10 (C) for linear kernels, $10 \times 10 \times 4 = 400$ ($C \times \gamma \times d$) for polynomial kernels and $10 \times 10 = 100$ ($C \times \gamma$) for sigmoid kernels, making a total of 510 possible combinations. For TiMBL, we allow 3 distance metrics, 4 feature weighting methods, 5 nearest neighbour values k and 3 voting strategies, good for a search space of $3 \times 4 \times 5 \times 3 = 180$ combinations.

6.2 Feature selection

The tasks of detecting suicide-related and severe messages in user-generated content are novel. As a consequence, we do not know from previous work what kind of information helps in disambiguating the categories. For this reason, we took an explorative approach to feature engineering. We constructed the various features outlined in Chapter 4 because they could have predictive power. With 46 feature groups and almost two million individual features, we impose few a priori restrictions on which features may or may not be informative.

It is unlikely, however, that using the full feature vectors will produce the best results. An ideal feature vector contains only highly informative features and no irrelevant ones, because these may negatively affect performance – especially when they are abundant and their cumulative mass drowns out the relevant features. This is why feature selection is important. The goal is to eliminate features that add little or no additional information beyond that provided by the other features.

Feature selection is similar to hyperparameter optimization in the sense that it can improve classifier performance considerably, and that it should be performed before comparing algorithms, given that some algorithms are more vulnerable to 'bad' features than others. There is also a practical motivation for feature selection: dimensionality grows with the number of features, as do the associated memory and CPU requirements.

Determining the informativeness of a feature is an important problem in machine learning. In Chapter 5, we already showed that both LIBSVM and TiMBL can apply a form of feature weighting, which promotes informative features in the model by giving each feature a real-valued weight depending on its relevance. Feature selection extends this idea, by removing features entirely, which would be equivalent to setting their weights to zero.

The feature selection methods that have been proposed in the literature can be divided into two groups: those where the selection is done independently of classifier performance (the *filter approach*), and those where classifier performance guides the selection (the *wrapper approach*) (Aha and Bankert 1996). We applied both approaches, as described below: feature filtering was first used to reduce the number of features, and the resulting feature set was further refined with a number of wrapped feature selection methods.

6.2.1 Feature filtering

With the filter approach for feature selection, an evaluation function is used to score each feature's informativeness for a given task, without explicitly testing the features with a learning algorithm. Selection can be done by keeping the n features with the highest score, or by removing features that score below a set threshold.

The evaluation functions used for filtering are statistical measures from the field of information theory (Cover and Thomas 2012). They estimate the relevance of a feature by calculating how good a predictor it is for the class label. Examples are *information gain*, *gain ratio*, and *chi-squared*, which are the three feature weighting metrics we use with TiMBL.

The information gain (IG) of a feature is the difference in entropy when the feature is present or absent. Entropy is a measure for the uncertainty in a random variable X. In this case, X is the unknown class label, which can take on two values (given binary classification). Its entropy $H(X)$, expressed in bits, is defined as

$$H(X) = -\sum_{x \in \{0,1\}} p(x) log_2 p(x) \tag{6.1}$$

where $p(x)$ is the prior probability of x. Entropy measures the minimum descriptive complexity of the class label given some features, and information gain quantifies the decrease in complexity when a feature is added:

$$IG_i = H(X) - \sum_{v \in V_i} p(v) \times H(X|v) \tag{6.2}$$

where V_i is the set of values for feature i.

Since information gain is sensitive to the number of values a feature can take, and has a tendency to overestimate the relevance of features with more values, a normalized version was proposed by Quinlan (1993), called gain ratio. It divides the information gain of a feature i by its *split info*, i.e. the entropy of its values:

$$GR_i = \frac{IG_i}{H(V_i)} \tag{6.3}$$

Chi-squared is a common statistical test that measures divergence from the expected distribution. It assumes that feature occurrence is independent of the class label. For a detailed description of its calculation, we refer to Daelemans et al. (2009).

A variety of other feature selection metrics exists, including *document frequency*, *mutual information*, *odds ratio* and *bi-normal separation*. The merits of these metrics have been empirically compared in benchmark studies, of which the ones by Yang and Pedersen (1997) and Forman (2003) are especially relevant to our work, because they benchmark against text classification datasets. The latter offers a comprehensive overview of twelve metrics that are compared against four optimization objectives (accuracy, precision, recall and F_1 score), on 229 binary text classification tasks with SVM.

Yang and Pedersen (1997) found that information gain and chi-squared allow aggressive feature removal with minimal loss in categorization accuracy. The findings of Forman (2003) corroborate that information gain is competitive, but chi-squared is said to behave erratically for very small expected counts, as is the case with our sparse bag-of-words vectors, and the skewed dataset where positive instances are relatively scarce. The bi-normal separation (BNS) metric introduced in the study outperforms most other metrics.

Based on these findings, we opt to use information gain. We could not consider the BNS metric for lack of a reliable implementation. Its promising performance for recall and on high-skew data offers an interesting alley for future work, however. Gain ratio, the improvement proposed for IG to correct the overestimated relevance of features with many values, was not tested in the aforementioned studies because there, all features were binary-valued. We ranked our features using both metrics. Notwithstanding the fact that gain ratio was expected to increase the relative importance of binary features, we found it to be overly harsh on real-valued ones. When ranking the features with gain ratio, only binary features appeared in the top n, even for values of n in excess of 100 000. As a consequence, real-valued feature groups (such as LSA-20 or EMO-ratio+) were effectively removed. Since we were interested in considerably reducing the feature set size (i.e. $n < 100\,000$), but also wanted to retain the diversity in the information sources, IG was deemed the more appropriate metric for filtering.

A threshold of 0.001 was used to filter features for the two tasks. Table 6.1 shows the impact on feature group size for both tasks.

Overall, we shrink the feature set size by 98.9% for the relevance task, and 99.5% for severity. This reduces the dimensionality by two orders of magnitude, and brings it to a level that can be handled computationally.

Feature group	unfiltered	relevance	severity
W1	51 670	981	323
W2	300 277	1 481	827
W3	475 167	787	970
LEM1	50 739	917	307
LEM2	289 438	1 460	797
LEM3	481 639	830	966
WCH2	3 383	773	298
WCH3	32 576	2 697	857
WCH4	136 426	5 218	1 430
LCH2	1 799	384	212
LCH3	18 136	1 836	701
LCH4	92 049	3 970	1 235
PAT-ratio+	1	1	n/a
PAT-ratio−	1	1	1
PAT-sum	1	1	1
DUO-ratio+	1	1	1
DUO-ratio−	1	1	1
DUO-sum	1	1	1
PAT-ratio+(last)	1	1	1
PAT-ratio−(last)	1	1	1
PAT-sum (last)	1	1	1
DUO-ratio+(last)	1	1	1
DUO-ratio−(last)	1	1	1
DUO-sum (last)	1	1	1
EMO-ratio+	1	1	1
EMO-ratio−	1	1	1
EMO-sum	1	1	1
EMO-count+	1	1	1
EMO-count−	1	1	n/a
TERM-exact	131	8	2
TERM-local	159	14	11
TERM-global	200	38	20
LSA-20	20	20	20
LSA-50	50	50	50
LSA-100	100	100	100
LSA-200	200	200	200
LSA-20-avg	1	1	1
LSA-50-avg	1	1	1
LSA-100-avg	1	1	1
LSA-200-avg	1	1	1
NE-presence	1	1	1
NE-count	1	1	1
NE-unique	1	1	1
LENGTH	1	1	1
CAPS-char	1	1	1
CAPS-token	1	1	1
Total ($n = 46$)	1 934 186	21 791	9 351

Table 6.1: Feature group sizes before and after applying information gain filtering for both classification tasks. Features with an IG value below 0.001 were removed.

The bag-of-words feature groups undergo the most drastic reductions. This is not surprising, given their sparsity: a large percentage of bag-of-words entries will occur only once or twice in the training data. We can therefore expect they can safely be eliminated, since their frequency suggests it is unlikely they will occur in future test instances. Likewise, very common words are typically uninformative because they are likely to appear in all posts, regardless of class.

Set size reduction in the other groups combined is mild at 48.5% for relevance and 51.8% for severity. All removed features are from the TERM groups, with the exception of PAT-ratio+ and EMO-count−, the only two feature groups to be filtered out entirely (for severity).

6.2.2 Wrapped feature selection

With wrapper methods, the second approach to feature selection, the informativeness of a feature set is determined by validating it with the intended learning algorithm. The basic idea is to try different feature sets (subsets of the entire feature space) and choose the one that gives the best validation results.

The main advantages of this approach are that it selects the optimal features for a specific problem and learner, rather than using a heuristic metric to estimate feature salience, and that it tests combinations of features rather than features in isolation, so that feature interactions and redundancies are considered.

As opposed to the aforementioned filtering methods, where individually scoring each feature takes linear time ($O(n)$ where n is the number of features), a wrapper method would take exponential time ($O(2^n)$) if an exhaustive feature subset search were performed. This disadvantage is compounded by the fact that evaluating a single combination (whereby a model needs to be trained and tested) is computationally much more involved than calculating a filtering metric. Exhaustive wrapped feature selection therefore quickly becomes unfeasible as n goes up.

Analogous to hyperparameter optimization, we are faced with a search problem, on which we will elaborate in Section 6.3. Even with an efficient search method, however, the number of individual features after applying the filtering method still makes for too large a space to be able to effectively search it. It therefore makes sense to do feature set selection instead, where features are bundled into a relatively small number of logical sets, which can then be selected or disabled as a whole. We define three ways of partitioning the features.

The most straightforward separation is by feature group. With 46 groups (44 for severity), this gives a reasonably sized search space. After *feature group* selection

experiments, the relevance of a feature group as a whole will be indicated by its selection status, and may allow to draw conclusions about which information sources are suitable for suicidality modelling.

In the *nbest* selection experiments, we limit the number of features in each group to 500, i.e. the 500 best according to the information gain metric. We include these experiments to test the potential benefit of having only the most informative features. As discussed above, it may be that large groups are at a selection disadvantage because they contain a portion of irrelevant features that negatively affect performance. With nbest selection, the number of groups remains the same, but the total number of features is reduced to 6 341 for relevance and 5 568 for severity.

Finally, we also test a more intricate separation method, *stratified* selection, whereby each feature group is sorted by IG and split into a number of bins, called *strata*. This allows finer-grained selection, the intuition being that strata at the low end of the IG spectrum might be removed.

Given the differences in feature group size, we vary the number of strata accordingly. We define the number of strata S_i for feature group i as a function of its feature count n_i, by rounding the cube root of group size to the nearest integer:

$$S_i = round(\sqrt[3]{n_i}) \tag{6.4}$$

The motivation for using the cube root is that it provides a good tradeoff between granularity and number of strata. Small feature groups ($1 \leq n_i < 100$) will be split into a small number of fine-grained strata ($1 \leq S_i \leq 5$). As feature groups sizes go up, they are split into more bins, but the number of bins grows slowly. This prevents the search space from becoming too large, but comes at the expense of granularity. A group with a 1 000 features, for example, will be split into 10 bins of size 100.

We obtain 187 stratified groups for the relevance task, and 154 for severity. WCH4, the largest feature group, is split into 17 bins (bin size 307) for relevance and 11 (bin size 130) for severity.

6.3 The search problem of optimization

As discussed above, hyperparameter optimization and feature selection each present a search problem that needs to be solved. They can be considered in sequence, for example by optimizing the hyperparameters on all features first, and then doing feature selection with the optimal hyperparameters. This approach ignores the interaction between features and hyperparameters. We

therefore perform *joint optimization*, where both problems are considered at the same time. While this approach is better from a theoretical point of view, it does make the search problem harder, due to the combinatorial explosion that follows when both search spaces are joined.

The two naive approaches to a search problem are *manual tuning* and *grid search*. Manual tuning is the process of iteratively testing a combination, checking the results, and testing with a different combination if the results are not satisfactory. Decisions about when to stop and which settings to change are left to the discretion of the researcher. While the simplicity of this approach makes it a popular solution when resources are scarce or for exploratory experiments, it is clear that it will not yield the best combination. Grid search, on the other hand, is guaranteed to find the optimal solution. This exhaustive search method tests every possible combination, and is therefore only feasible when the search space is small.

Hillclimbing has long been a popular search procedure, particularly for feature selection. Depending on the implementation, it starts with an empty feature set (for *forward selection*), the full feature set (for *backward elimination*) or some randomly initialized feature or hyperparameter set (for *bidirectional hillclimbing*). Next, all neighbouring states are tested (by adding or removing one feature, or changing one parameter), and the one that achieves the highest increase in performance is selected. This process is repeated until a state has no superior neighbours. The final state is considered the optimal solution.

Although hillclimbing is efficient in finding an optimum, it is not guaranteed to find the best solution in the search space. After initialization, it follows the direction of the upward slope, and will converge on the nearest local optimum, which may differ from the global optimum. Whether or not the global optimum is found therefore depends on the initialization.

A number of alternative methods exist that provide more effective search, such as simulated annealing (Kirkpatrick et al. 1983), local beam search (Russell and Norvig 1995) and genetic algorithms. We opted to use the latter, which has been successfully applied for many NLP tasks. Contrary to the hillclimbing approach, optimization runs using *genetic algorithms* are initialized from a variety of points in the search space, and upward slopes are not followed deterministically. In the next section, we discuss the theory of genetic algorithms, and how we integrated them in our experimental setup.

6.4 Genetic algorithms

6.4.1 Theory

In large search spaces where exhaustive search is computationally not feasible, genetic algorithms (GAs) have for a long time been used to find optimal or near-optimal solutions. Below, we provide a high-level overview of how they work. For further details on the theoretical basis and functioning of GAs, we refer to Holland (1975), Goldberg (1989) and Whitley (1994).

Genetic algorithms are search methods inspired by evolutionary biology. In essence, the search problem is likened to how in Darwinian evolution theory, natural selection is an ongoing search for the genetic traits that offer the best chances of survival and reproduction, the so-called *survival of the fittest*.

Natural selection occurs in a population of individuals. Individuals share a *genome*, the genetic blueprint of the organism, but they can differ in *genotype*, the specific combination of *alleles* for each gene in the genome. An allele is the 'value' for a gene, which results in a specific trait. This genetic diversity is introduced in part through *mutation*, random changes in alleles, and also through reproduction, where the new genotype of a child is constructed by combining segments of the genotypes of the parents, a process called *crossover*.

The basic principle of natural selection is that when a genetic trait makes an individual more successful (in terms of *fitness*, the ability to survive and reproduce in a given environment), it will have a reproductive advantage over individuals without it. As a result, a successful allele will have a higher probability to be passed on, causing its frequency in the population to increase across generations. Evolution moves the population towards better fitness, or if viewed as a search problem, it produces individuals that increasingly approach an optimization target.

Search problems as genomes

Genetic algorithms borrow the concepts of fitness-based selection, mutation, inheritance and evolution, and apply them to a search problem. First, the search space is represented as a genome of fixed length. In the case of joint feature selection and hyperparameter optimization (see Figure 6.1), the genome will consist of one binary-valued gene for each feature group (a bit with two alleles: 1 if the feature group is selected, 0 if it is not), and one multi-valued gene per hyperparameter (e.g. the various values for k nearest neighbours in TiMBL).

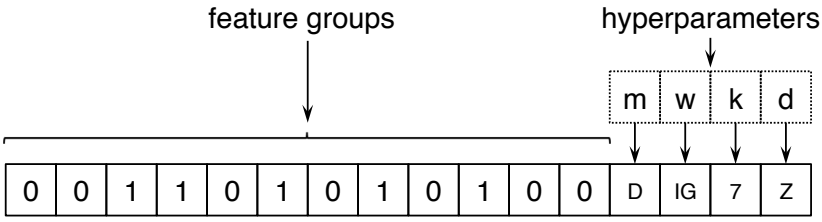

Figure 6.1: A potential solution to the problem of joint optimization in TiMBL, coded as a genotype. Its structure is dictated by the genome, the genetic representation of the search space.

The general procedure a genetic algorithm follows during an optimization run is illustrated in Figure 6.2.

Initialization

First, an initial population is created, containing a fixed number of individuals. Each individual is a random instantiation of the genome, i.e. a genotype of which all alleles have been set randomly, that represents a candidate solution to the optimization problem. The size of the population determines the amount of variation it can contain, so a larger population size increases the probability of finding the global optimum, but will be computationally more expensive.

Fitness

Next, the fitness of all individuals in the population is evaluated. This is done with a *fitness function*, an essential part of the genetic algorithm. It assesses the quality of a given individual for solving the problem at hand. In our setup for joint optimization, the fitness is determined by validating the learner that corresponds to the genotype of the individual. In other words, we configure a TiMBL or LIBSVM model to use only the selected features and hyperparameters, run tenfold cross-validation on the training data, and use the resulting F-score value as the fitness score.

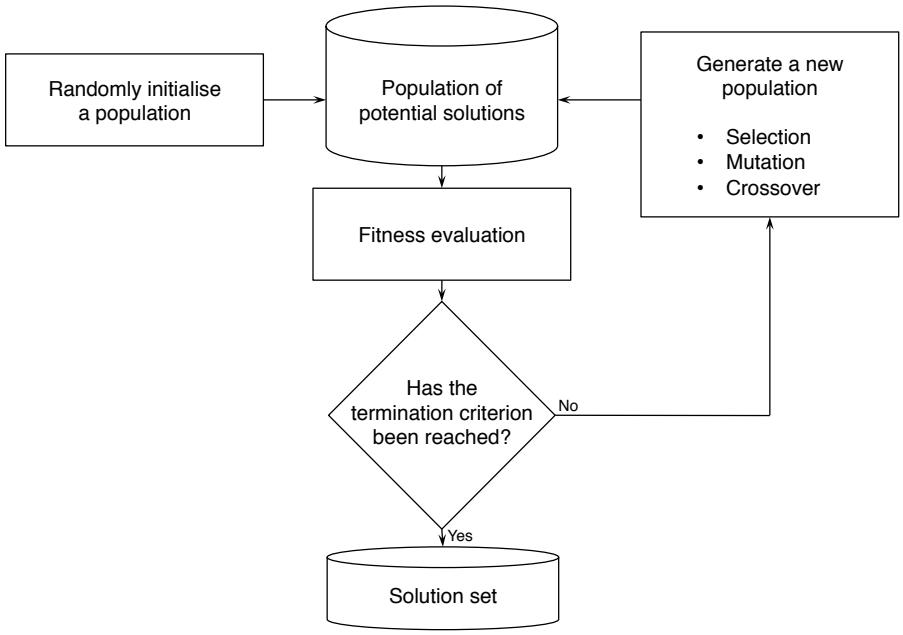

Figure 6.2: Graphical representation of an optimization run with a genetic algorithm.

Termination

When a generation of individuals has been evaluated, the genetic algorithm checks whether the stopping criterion has been satisfied. Typically, there will be a set number of generations that the algorithm is allowed to evaluate, so that runtime is limited. The GA run can also converge on an optimal solution before the maximum number of generations has been reached. We consider a solution to be optimal when the highest fitness in a population (the score of the *elite* individual) has remained the same for five generations. Ideally, the maximum number of generations is set sufficiently high for the GA to terminate on an optimal solution instead.

Selection

If the stopping criterion has not been met, a new population of individuals is created, called the *offspring* of the previous generation. In this phase, a selec-

tion method determines which individuals in the parent generation will survive and produce offspring. Selection occurs on the basis of fitness, but can be implemented in various ways (Goldberg and Deb 1991). Popular techniques are *roulette wheel selection*, where the probability that an individual is selected is proportionate to its fitness divided by the sum of fitnesses, and *truncation selection*, where individuals are sorted from best to worst and only the n best individuals are selected as parents. In *tournament selection*, a fixed number of individuals is randomly picked from the population to compete in a 'tournament', where an individual's probability of winning is proportionate to its fitness. The winner is selected as a parent. This process is repeated as many times as there are individuals to be selected. Selection pressure depends on the tournament size: when more individuals are picked to compete in a tournament, weak individuals have a smaller chance to be selected.

Reproduction

After selection, we have an intermediate population of parents that will spawn the next generation. There are two mechanisms at work during reproduction that introduce genetic diversity: mutation and crossover.

During mutation, a parent's genotype can be altered. Each gene has a probability of being mutated, whereby its allele is changed to some random other value (see Figure 6.3). In the case of binary genes, the allele is flipped. The probability of mutation is fixed, typically at a moderate value so that mutation produces subtle changes rather than drastically altered genotypes.

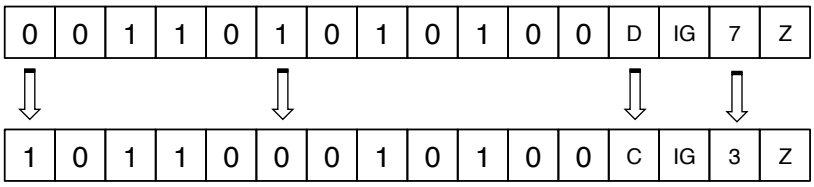

Figure 6.3: Example of mutation applied to an individual, where 4 genes have been randomly changed.

Crossover, on the other hand, randomly exchanges segments of the genetic material of two parents and combines them into two new genotypes. Figure 6.4 provides an illustration of *single point crossover*: the parental genotypes are cut at the same point in the genome, and the segments from that *crossover point* onwards are swapped. Genetic recombination happens with a certain crossover

probability, varying between 0 (no crossover) and 1 (crossover always applies). It is of note that with crossover, the order of genes in the genome becomes important. Grouping related genes (e.g. hyperparameters, or features derived from a common information source) in the genome may allow the GA to create so-called *building blocks*: low-order gene combinations with above-average fitness (Goldberg 1989). With random genome ordering, crossover can more easily disrupt the formation of such high-performance substrings.

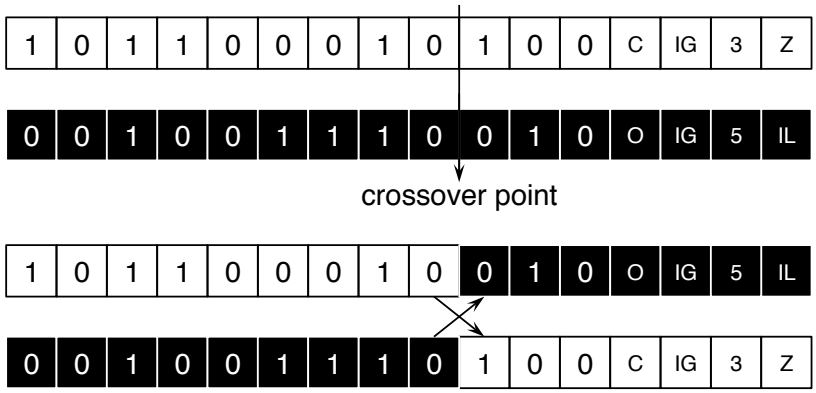

Figure 6.4: Example of single point crossover, where the genotypes of two parents are combined to produce two children.

Crossover and mutation have different roles in the search for an optimal solution. Crossover is explorative: it allows the search to make big 'jumps' to new areas that have not been explored before. When the genetic material of two parents is crossed over, the resulting genotype will search an area that lies somewhere in between the two parent areas. This exploration of the search space enables the GA to gain information on the problem and helps it to avoid overlooking optima.

Mutation, on the other hand, is exploitative. Because it only makes subtle changes, it allows the search to be optimized within a promising area. It exploits the information of the (successful) parent by making small diversions in its vicinity. Mutation is also the only way of introducing unseen alleles or changing allele frequencies in the gene pool, since crossover can only recombine alleles from the previous generation.

The two mechanisms have competing objectives, but for most search problems, their effects are complementary. Using only crossover, the search cannot converge on an optimum. GAs that only use mutation should eventually converge,

but they will be slow in exploring the search space. It is therefore generally recommended to use both.

Instead of having all parents reproduce, it is possible to keep some of them unchanged, so as not to remove potentially optimal solutions. This is known as *elitism*, whereby the most fit parents are guaranteed a place in the next generation without undergoing mutation. This does not prevent them from also being selected as parents for mutation and crossover. Elitism is a strategy that encourages convergence.

Solution

After reproduction, the fitness of the resulting generation is evaluated, and the evolution continues until termination.

The solution of the search is the set of individuals in the final generation that share the best fitness score. There can be more than one solution when the GA has found multiple optima in the search space, or when individuals with different genotypes result in the same model (the same *phenotype*), because certain alleles may have no impact.

6.4.2 Implementation

With genetic algorithms, we have at our disposal a means of finding solutions in a large search space. It is much more efficient than e.g. testing every possible solution, but the computation time t required to evaluate a single candidate solution is still quite significant. On a single 3.5 GHz core, for example, doing tenfold cross-validation on our experimental dataset (with all features and default hyperparameters) takes in the order of hours for LIBSVM and tens of hours for TiMBL.

It should therefore not surprise that fitness calculation is the most time-consuming step in a GA search. Given a population size m and a maximum number of generations n, running a GA optimization sequentially (on one core) would require a maximum of $m \times n \times t$ time. In practice, the required time can be less when the GA converges before in fewer than n generations. Even with conservative parameters (e.g. $m = 50$ and $n = 20$), the maximum computation time is in the order of months to years. This prohibits the use of moderate to large values for m and n.

If all the individuals in a population are evaluated simultaneously, the maximum computation time is reduced to $n \times t$, which is in the order of days to weeks. If the

fitness calculation involves k-fold cross-validation, further speedups are possible by evaluating all the folds in parallel. This can reduce the computation time t to a theoretical[1] minimum of $\frac{t}{k}$. The overall computation time can therefore also be reduced by up to a factor k.

6.4.3 Distributed GA optimization with Gallop

The genetic algorithm toolbox Gallop (Desmet and Hoste 2013b) was developed to run the type of massively parallel optimization experiments described above. It is a Python library based on DEAP[2], the Distributed Evolutionary Algorithms in Python framework (Fortin et al. 2012).

Gallop provides the functionality to wrap a complex optimization problem as a genome, and to distribute the computational load of the GA run over multiple processors or to a computing cluster. It is specifically aimed at problems involving natural language.

As explained above, the problem of hyperparameter optimization can be represented in a genome with one multivalued gene per hyperparameter. Gallop currently includes wrappers for TiMBL, LIBSVM and CRF++[3], three learning algorithms that have been used extensively in NLP research. Each hyperparameter of these learners is exposed to the user, who can either choose to use the default (full) range of options, reduce or increase the set of options manually (e.g. allow only two kernels for SVM, or increase the range of values for its C parameter), or override the optimization of that hyperparameter by using a fixed value. When a population is created or offspring produced, Gallop builds genotypes with the available hyperparameter options, and checks them for compatibility. Incompatible options are disabled. With a linear SVM kernel, for example, the γ and d hyperparameters are removed.

The problem of feature selection is more involved. Gallop supports individual and grouped feature selection, and selected features or feature groups are represented as bits in the genome. TiMBL includes a command line option to ignore specific features, which makes it easy to do feature selection without having to rewrite the dataset. LIBSVM and CRF++ lack this feature, so for each feature selection combination, Gallop needs to rewrite the dataset with all disabled features removed.

[1] The theoretical minimum can be achieved if all folds take an equal amount of time, and there is no sequential overhead before or after the parallelized computation. In our experiments, the fitness calculation will take as long as the slowest fold, plus a minimal overhead.

[2] http://deap.gel.ulaval.ca/

[3] http://crfpp.googlecode.com/

A variety of file formats is supported: C4.5, ARFF, sparse datasets, and formats that include sentence and document boundary information (typically used for sequence tagging tasks such as part-of-speech tagging or named entity recognition). Gallop ensures format compatibility with the learner, and implements the necessary methods for feature selection rewriting. Datasets can be shuffled, split into a fixed train and test partition, or split into random or balanced folds for cross-validation. Information about sentence or document boundaries is used to avoid splitting them across folds.

To evaluate an individual, Gallop prompts the learner to train a model and predict the labels on the test set via system calls, and parses the results to obtain a list of labels. In case of cross-validation, this procedure is repeated for every fold, and can be parallelized to a configurable number of cores. Fitness can then be calculated with internal metrics (e.g. accuracy, precision, recall or F-score with a specified β value, using micro- or macro-averaging), or with an external scoring script (e.g. the `conlleval` script used for the CoNLL shared tasks on named entity recognition).

The top-level GA process is implemented in the DEAP framework. It keeps track of the current population and its history (so that identical individuals are only evaluated once), and handles selection and reproduction. Gallop inherits its parameters with regard to population size, selection settings, mutation and crossover rate, stopping conditions, etc. The population history is stored after each generation. This allows for *checkpointing*, resuming the GA run after an error or restarting it with different termination settings.

When Gallop is configured to run a GA optimization experiment on a single server, the individuals in a population are either evaluated sequentially, or in parallel using a specified number of cores. Because the number of individuals in a generation (e.g. 100) will typically be higher than the amount of cores on a server (e.g. 8), the evaluation will happen in a staggered fashion, rather than entirely in parallel. For this reason, Gallop can also be run on a supercomputer, i.e. a high performance cluster consisting of many worker nodes. Each generation is then submitted as an array of job requests to be processed simultaneously, and Gallop polls the cluster until all jobs are finished. The computational resources (Stevin Supercomputer Infrastructure) and services used in our work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, the Hercules Foundation and the Flemish Government (Department EWI)[4].

[4]https://vscentrum.be/en

Experimental Gallop settings

Setting the parameters for a GA is a search problem in itself, one that has given rise to a dedicated field of study (Lobo et al. 2007). Despite efforts to find 'generally optimal' parameter values that work for a wide range of optimization problems, it is currently acknowledged that specific problem types require specific GA settings, and that the quest for universal near-optimal settings is lost a priori (Michalewicz and Schmidt 2007). We consider GA parameter optimization to be outside the scope of this dissertation, but attempt to make experimental parameter choices that are justified by heuristics and insights from the literature.

The population size was set to 100. We kept the size at the low end of what is generally recommended, since our validation procedure is computationally very expensive. We used single-point crossover with a probability of 0.9, and a mutation rate to 0.3. These settings are both relatively high. The latter promotes exploration, which can compensate for the small population size and avoid premature convergence.

We applied elite selection at a rate of 0.1, i.e. promoting the fittest 10% of a population to the next generation. For the remaining 90%, we used tournament selection with a tournament size of three. Roulette wheel selection has been shown to be significantly slower than other methods, and truncation selection offers little exploration (Goldberg and Deb 1991). Tournament selection provides a good trade-off between speed and exploration when the tournament size is sufficiently small.

Evolution was stopped after 50 generations, or when the best fitness has changed less than 0.0001 over the last 5 generations. In practice, all optimization runs converged before reaching the maximum number of generations.

6.5 Summary

This chapter described our model optimization methodology, aimed at finding a good combination of algorithm settings and relevant features. We performed feature filtering based on information gain, reducing the number of bag-of-words features by 99% and other features by 50%. Three methods were defined for wrapped selection: regular, nbest and stratified feature group selection.

To solve the search problems of hyperparameter optimization, feature selection and joint optimization, we experiment with genetic algorithms. Given the computational cost of running such experiments, we use Gallop, a GA imple-

mentation that allows distributed computing on the HPC supercomputer infrastructure. The experimental results are presented and discussed in the next chapter.

CHAPTER 7

Experiments

In the previous chapters, we reviewed the methods used to obtain the necessary parts for an experimental model to detect user-generated suicidal text content. We discussed the various components (data, annotation, information sources and machine learning algorithms) and the methodology for validating and optimizing models. In this chapter, those elements are combined into a series of experiments.

We start by introducing the experimental setup common to all experiments in Section 7.1, which covers data sampling, the baseline classifier and *k-nearest fitness*, a method used for discussing optimization results. Next, we go on to presenting and discussing the results for the experiments on direct relevance and severity classification (7.2 and 7.3), cascaded severity classification using gold-standard and predicted relevance labels (7.4). Section 7.5 presents the scaling experiments and a qualitative analysis, and we summarize our experimental findings in Section 7.6.

7.1 Experimental setup

7.1.1 Corpus sampling

In Chapter 3, we described the text corpora that were collected and annotated. For the experiments, two corpora are available with messages from the social media platform Netlog. The first contains suicide-related material and was entirely annotated. After duplicate removal and cleanup, it consisted of 1 040 posts, 851 of which were annotated as relevant, and 257 as severe. Secondly, we have a reference corpus of 373 349 Netlog posts, which were not annotated.

From these corpora, we sampled a number of subsets for the experiments.

Development set

All the models were trained and tested on a 10 000 post development set, using 10-fold cross-validation. The majority of the annotated posts was included in this corpus, to provide sufficient positive instances for learning. We removed 40 posts from the suicide corpus for the held-out testset, described below. The remaining 1 000 posts were added to the development set, which then contained 811 relevant posts and 237 severe ones.

The remaining 9 000 posts were sampled from the reference corpus. Section 3.1.2 described a spot check for suicide-related material in the reference corpus, which was performed on this sample and did not produce any hits. We therefore consider labeling all posts as negative to be a sufficiently precise assumption.

The resulting development set exhibits significant class skew. Some 8% of the instances is positive for the relevance task, and 2.4% for the severity task. Since we do not optimize towards accuracy, we do not consider this skew to be problematic. Moreover, randomly sampled user-generated content would be skewed much more towards the negative class.

When sampling from the reference corpus, another consideration with regard to skew had to be addressed. Compared to the suicide corpus, the reference corpus is skewed heavily towards very short posts (cf. Figure 7.2). If subsets were to be sampled uniformly, the average reference post would easily stand out for the classifier because of its size. We therefore used stratified sampling, a technique to create a sample that mimics the post length distribution in the suicide corpus.

First, a sorted array was created with the length in characters of each post in

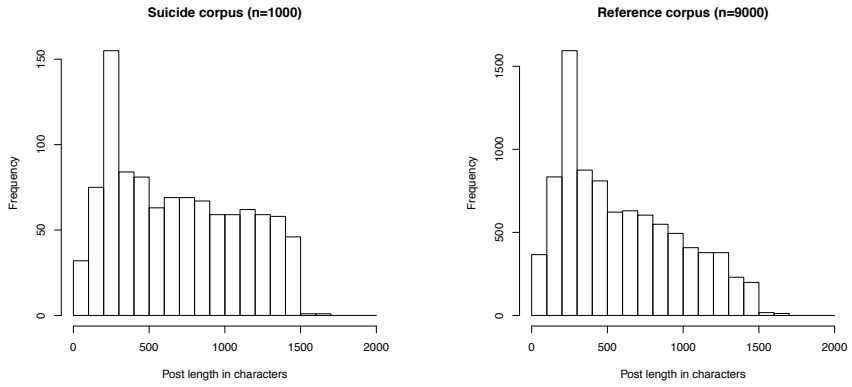

Figure 7.1: Histograms of the post length distribution in the two constituent parts of the development corpus: the 1 000 post sample from the suicide corpus on the left, and the 9 000 post stratified sample from the reference corpus on the right.

the suicide corpus. We split the array into bins with an interval of one hundred characters, yielding the distribution shown in the left-hand histogram of Figure 7.1. The reference corpus was binned in the same manner. For each post to be sampled, the aforementioned distribution was used as a probability function to determine the bin to randomly pick from. In this manner, we proportionately sampled the 9 000 posts. The right-hand histogram of Figure 7.1 already displays a slight skew towards shorter posts: due to the small number of long posts in the reference corpus, the right-most bins were exhausted.

Held-out testset

In Section 5.3, we already mentioned a drawback of using cross-validation. The procedure of rotating the training and test folds ensures that models are never trained on the test fold, but since the bag-of-words features have been engineered using all the data together, the test folds are not entirely blind.

Cross-validation results offer a valid means of comparing models amongst each other. This makes the technique very suitable for optimization, and for observing the differences between the various learning algorithms, or the impact of features and hyperparameters. However, cross-validation results alone do not offer an accurate picture of a model's expected performance on unseen data: not only because the test data is not fully blind, but especially since the scores

are obtained after optimization.

This is why we also report scores for each model on a held-out testset, which is completely independent from the data used for model training and optimization. We sampled 40 relevant posts from the suicide corpus, half of which were severe. It should be noted that this sample does not replicate the $1 : 3.5$ ratio of severe versus relevant posts found in the fully annotated dataset. The $1 : 2$ ratio in the testset is motivated by the aim not to reduce the training set too much, whilst ensuring a minimum amount of severe instances.

A stratified sample of 10 000 reference posts was added to the positive instances. The post length distribution, shown in the top left of Figure 7.2, is similar to that of the reference sample in the development corpus, although the proportion of long posts is further reduced.

We use the held-out testset to benchmark model performance on very skewed unseen data (0.4% incidence for relevance and 0.2% for severity), and for qualitative error analysis.

Datasets for scaling experiments

The aim of this study is to develop models capable of separating suicidal posts from a vast pool of unrelated material. If a model flags many irrelevant posts for review, prevention workers may be overwhelmed by the amount of noise. We therefore do scaling experiments to observe the behaviour of the best models on large datasets, and to get an impression of their practical usability. To that end, we sampled increasingly large subsets (of 20 000, 70 000 and 200 000 posts) from the remainder of the reference corpus, with a progressive decline in average post length (cf. Figure 7.2).

The held-out testset ($n = 10040$) was incrementally enlarged with these subsets, and we thus obtained scaling testsets of around 30 000, 100 000 and 300 000 posts. The percentage of (known) positive instances is very small and reflects the distribution that would be encountered in real-world user-generated content.

7.1.2 Experimental setup per task

We performed three large sets of experiments to tackle the tasks of relevance and severity classification. For the detection of suicide-related posts, we test *direct relevance classification*, in which a classifier's objective is to label a post as either relevant or not. Recognizing posts written by (or about) persons with a severe risk of suicide is more complex. This is why we test both *direct severity*

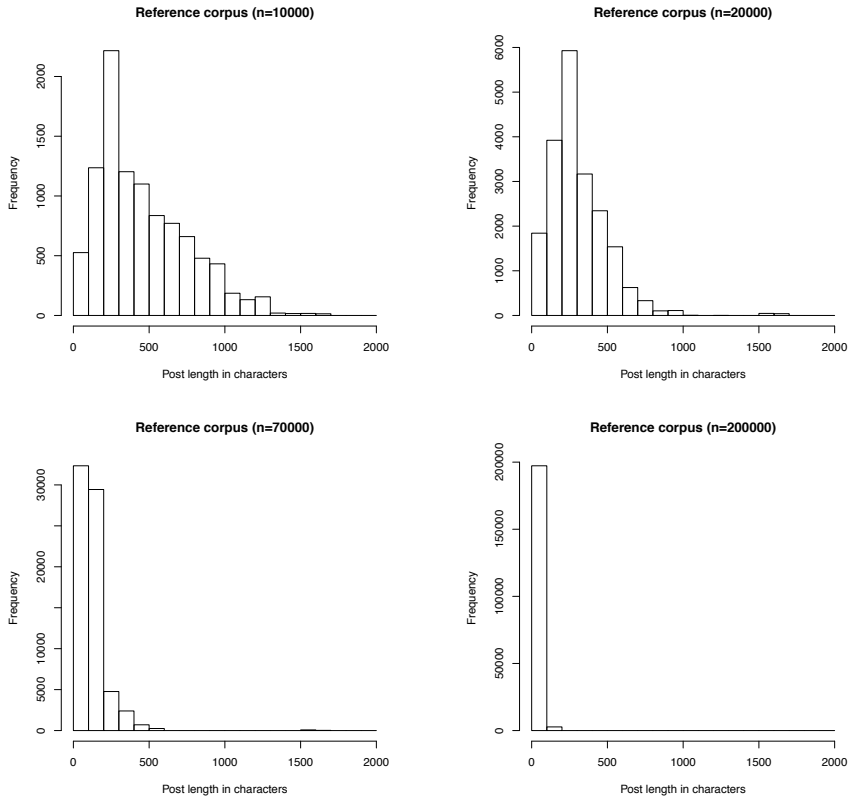

Figure 7.2: Histograms of the post length distribution in four stratified samples from the reference corpus: the held-out test set (top left), and the three scaling sets of size 20 000, 70 000 and 200 000.

classification and *cascaded severity classification*. With the latter, a classifier receives the suicide-related posts detected by a relevance model, and needs to determine whether or not it contains a severe suicide threat.

In order to best achieve the detection objectives, given the available information sources and learning algorithms, we performed model optimization with Gallop. It was used to optimize towards F_1, our main performance metric, and separately towards F_2, a variant that places greater emphasis on recall, so that we can compare the impact of both fitness functions. In each set of experiments, the four feature selection strategies (using all features, i.e. no selection, or full, nbest or stratified feature groups selection) were combined with the option to

do hyperparameter optimization or not. This results in one unoptimized run (for which no GA was required), one with hyperparameter optimization only, three with feature selection only, and three joint optimization runs.

In the direct tasks, the validation of one individual requires classifying the entire set of 10 000 instances. Validation for the cascaded tasks, by contrast, only involves a subset of instances: the ones that had been labeled as relevant in the previous stage. Optimization experiments with TiMBL on the full dataset could not be carried out, because validation of an individual exceeded the maximum allowed time on an HPC worker node (72 hours). Consequently, we only used LIBSVM for the direct classification tasks, and both TiMBL and LIBSVM for the cascades.

For each Gallop optimization run, we report the scores of the best individual. As explained in Section 5.3, scores on the development set are obtained by doing tenfold cross-validation. On the held-out testset, on the other hand, we calculate scores based on the predictions from a single model trained on the full development set. We report precision, recall, F_1 and F_2, and focus on the F-score variant that was used as the fitness objective.

7.1.3 k-nearest fitness

At the end of a GA optimization run, the highest fitness score may be shared by multiple individuals that have different features or hyperparameters (for an explanation, see Section 6.4.1). Runner-up individuals to that elite may also be considered valuable solutions to the search problem, since small differences in fitness can be caused by idiosyncrasies of the validation process (e.g. the properties of the dataset).

When discussing the results of a GA experiment, we may therefore refer to the *k-nearest fit* solution set. These are the individuals that obtained one of the top k fitness scores, given an arithmetic precision (e.g. by rounding the scores to four decimal places).

Table 7.1 gives an example pool of ten raw fitness scores from the individuals of a GA run, and illustrates how the values for k and precision affect the k-nearest fit solution set. We will use a precision of four significant figures throughout this chapter, and set k to three. In the example, this produces a solution set with five individuals.

ranked fitness scores	$n = 3$, $k = 5$	$n = 4$, $k = 3$
0.973582217871	0.974	0.9736
0.973582217871	0.974	0.9736
0.973545007264	0.974	0.9736
0.973410471038	0.973	0.9734
0.971700462588	0.972	0.9717
0.970603799112	0.971	
0.970330731929	0.970	
0.970309588910	0.970	
0.968805437036		
0.968114960616		

Table 7.1: Two examples of the k-nearest fit individuals, with different values for precision (to n significant figures) and k.

7.1.4 Baseline classifier

A straightforward baseline classifier was implemented, against which to compare the models' performance. It labels a post as positive if the keywords *zelfmoord* or *zelfdoding* appear in it, either as separate words or in compounds. It thus predicts the same labels for the relevance task as for the severity task.

task	**dataset**	$\mathbf{F}_1$	$\mathbf{F}_2$	**precision**	**recall**
relevance	development	85.89	86.74	84.51	87.32
	held-out	91.89	87.63	100.00	85.00
severity	development	36.21	54.44	23.24	81.93
	held-out	59.26	70.18	47.06	80.00

Table 7.2: Baseline scores on the development and held-out datasets, for the relevance and severity tasks.

Table 7.2 displays the baseline scores for both tasks, on the development and held-out datasets. We find that the baseline achieves high recall scores on both tasks and datasets. For the relevance task, precision is also high, especially on the held-out dataset, where all posts containing one of the search terms are relevant. Logically, precision is much lower for severity, since the baseline does not make a distinction between relevant and severe posts.

It should be noted that part of the reason why the baseline performs so well in terms of recall is that it is informed about the way positive instances were obtained. Since we had to resort to using keywords to obtain sufficient suicide-

related material for annotation, our dataset does not include many posts that present more implicit mentions of suicide. As a consequence, the scores of the baseline can be considered an overestimation: its naive hypothesis pays off particularly well on our dataset, but we can assume that its recall would be much lower on a more realistic dataset.

7.2 Relevance classification

In this section, we investigate the viability of our approach for relevance classification, the task concerned with detecting posts about suicide among unrelated posts. Given the number of features (21 791) and instances (10 000) for this task, we only tested classification with SVM, because the time and memory requirements of TiMBL were impractical with our setup.

7.2.1 Cross-validation results

Tables 7.3 and 7.4 list the results of two sets of experiments on the development dataset. For ease of reference, we display the relevant baseline results (for this task and dataset) in the first row. The second row shows the results obtained with a LIBSVM classifier configured to use the default hyperparameters and all features, i.e. the unoptimized results. The next seven rows each represent a separate Gallop optimization run, with various optimization settings: with or without hyperparameter optimization (HO) and with none or one of the three feature selection (FS) strategies. For these optimized runs, we display the scores of an elite individual, i.e. a classifier with settings optimized towards a particular fitness score. The classifiers in Table 7.3 were optimized towards F_1, those in Table 7.4 towards F_2.

On each row, we display the four metrics discussed in Section 5.3: precision, which measures the amount of noise in the predicted positives, recall, the *hit rate* to gauge the number of missed positives, and F-score metrics which combine the two (balanced F_1 and weighted F_2 to increase the importance of recall). All metrics are formatted as percentages. The best value for a particular metric is boldfaced, and classifier scores that are below baseline performance are in grey.

For a graphic representation of the scores, we refer to Figure 7.3. It displays the absolute scores in terms of F_1 or F_2, depending on the optimization objective, and the percentage-wise error reduction that is obtained after optimization, as compared to the unoptimized classifier.

HO	FS	F_1	F_2	precision	recall
baseline		85.89	86.74	84.51	87.32
no	none	90.61	90.48	90.84	90.39
	group	91.30	91.20	91.47	91.13
	nbest	91.02	90.72	91.53	90.52
	strata	92.39	**92.59**	92.05	92.73
yes	none	91.50	91.50	91.50	91.50
	group	91.95	92.49	91.06	**92.86**
	nbest	92.32	92.42	92.15	92.49
	strata	**92.55**	**92.59**	**92.50**	92.61

Table 7.3: Relevance classification scores on the development set, optimized towards F_1 (HO = hyperparameter optimization, FS = feature selection).

HO	FS	F_1	F_2	precision	recall
baseline		85.89	86.74	84.51	87.32
no	none	90.61	90.48	90.84	90.39
	group	91.31	91.28	91.37	91.26
	nbest	91.04	90.87	91.33	90.76
	strata	92.59	92.89	**92.08**	93.10
yes	none	91.47	92.81	89.32	93.72
	group	92.31	93.22	90.82	93.84
	nbest	92.27	**93.35**	90.52	**94.09**
	strata	**92.69**	93.31	91.69	93.72

Table 7.4: Relevance classification scores on the development set, optimized towards F_2 (HO = hyperparameter optimization, FS = feature selection).

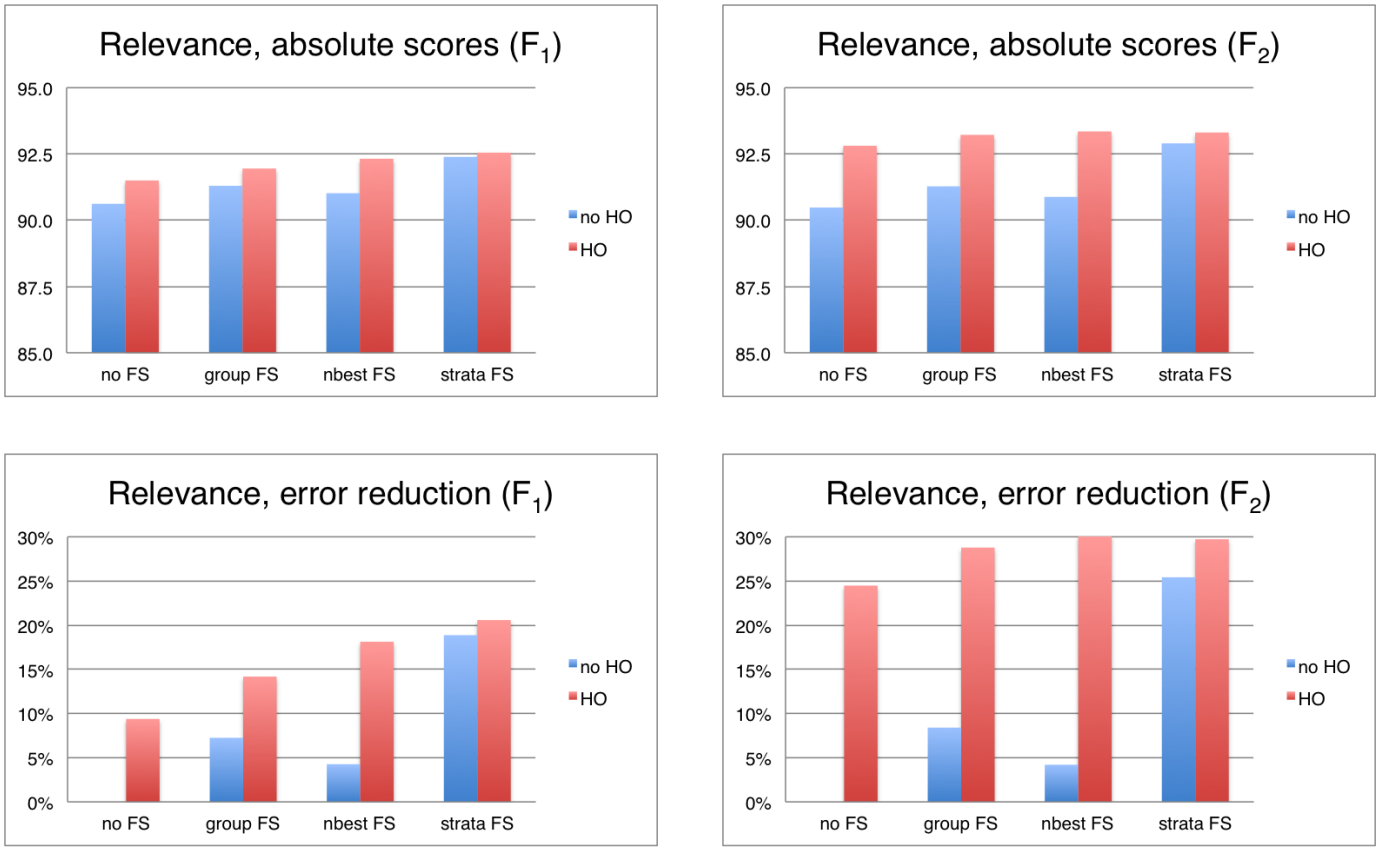

Figure 7.3: Absolute scores (top row) and percentage-wise error reduction after optimization (bottom row), for the relevance task on the development set. Scores are expressed in terms of F_1 (left) or F_2 (right), depending on the optimization objective.

A first observation to make is that the unoptimized system beats the baseline on all scores, with an F_1 of 90.61% and F_2 of 90.48. In consequence, all optimized classifiers also outperform the baseline.

Every optimization search with genetic algorithms results in improved scores. This is true regardless of the fitness objective, and is readily discernible in Figure 7.3. In isolation, hyperparameter optimization reduces the error for F_1 and F_2 by almost 10 and 25 percent, respectively. The large performance gains observed for F_2 indicate that hyperparameters have a significant impact on recall. SVMs have a good reputation with regard to robustness to bad features, because of their effective internal feature weighting. Nevertheless, we find that feature selection is equally worthwhile. Doing only regular or nbest feature group selection brings modest error reductions (around 5%), but the effect of stratified feature group selection matches or exceeds that of hyperparameter optimization, at 20% for F_1 and 25% for F_2.

With joint optimization, the differences between the three feature selection techniques are less outspoken. Overall, the highest result for F_1 is obtained when hyperparameter optimization is combined with stratified feature selection (92.69%), and for F_2, when it is combined with nbest feature selection (93.35%). It is of note that the best F_1 value is obtained in an optimization towards F_2, although the difference with the best score when optimizing towards F_1 (92.55%) is small, and is obtained with the same stratified setup. We see the same phenomenon with many other F_2 classifiers that match or beat the F_1 performance of their F_1-optimized counterparts. Ideally, we would expect the F_1 optimization runs to produce the highest F_1 scores. The fact that this does not happen can be explained by the non-deterministic nature of genetic algorithms: no two runs will be the same, and there is no guarantee that they will converge on the same near-optimal solutions. To further increase the chances of finding the optimal solution, we would need to change the GA settings to search more (bigger population) and longer (stricter termination criteria). In Section 7.4.1, we describe a GA experiment where such settings are tested.

Another explanation could be that optimizing towards recall is the better strategy for this task. All classifiers optimized for F_1 obtain a score that is balanced in terms of precision and recall: the difference between both terms rarely exceeds one percentage point. When comparing the F_1 classifiers one-to-one to the F_2 classifiers, we find that the latter consistently achieve lower precision and higher recall. That is to say, the different optimization objectives reliably steer the GA in the preferred direction. The aim for better recall eventually leads to the best F_1 scores as well, and it is plausible that sub-optimal solutions with high recall were discarded in the F_1 optimizations before they could be fine-tuned for better precision.

7.2.2 Results on the held-out testset

Next, we benchmark the performance of the different models on the skewed held-out dataset. The results of these experiments are shown in Tables 7.5 and 7.6.

The results show very strong baseline performance, at 100% precision and 85% recall. As was discussed in Section 7.1.4, our held-out dataset is particularly inducive to such strong performance, with 34 out of 40 positive instances containing one of the baseline keywords, all of which happen to be relevant. The baseline does not find the six positive instances that do not contain a keyword.

None of the hypotheses formed by the trained models are as simplistic. As opposed to the held-out set, the development set does contain irrelevant posts with a keyword (hence, the baseline could not achieve 100% precision there). Both datasets contain positive instances without a keyword.

All models match or improve the baseline performance for recall: four to six positive instances are missed (for a qualitative analysis of these results, see Section 7.5). When these recall scores are combined with high precision, the models outperform the baseline both on F_1 and F_2.

It is noteworthy that the models which underwent hyperparameter optimization achieve much higher precision on the held-out testset (with differences around fifteen percentage points). This trend was not apparent on the development data. Overall, however, we observe the same trends on both datasets: hyperparameter optimization is beneficial, and stratified feature group selection yields strong models.

In conclusion, these results reveal that we can successfully model the relevance task for an application where a good balance between precision and recall is important. In spite of the high skew in the data, precision is very good, so the amount of noise in the results is minimal. The effort of optimizing models is rewarded with a substantial error reduction, both on development and test data.

HO	FS	F_1	F_2	precision	recall
baseline		91.89	87.63	**100.00**	85.00
no	none	84.34	86.21	81.40	87.50
	group	83.34	85.79	79.55	87.50
	nbest	80.00	82.93	75.56	85.00
	strata	81.40	84.95	76.09	87.50
yes	none	92.31	90.91	94.74	**90.00**
	group	90.91	88.83	94.59	87.50
	nbest	91.14	90.45	92.31	**90.00**
	strata	**93.51**	**91.37**	97.30	**90.00**

Table 7.5: Relevance classification scores on the held-out testset, optimized towards F_1 (HO = hyperparameter optimization, FS = feature selection).

HO	FS	F_1	F_2	precision	recall
baseline		91.89	87.63	**100.00**	85.00
no	none	84.34	86.21	81.40	87.50
	group	83.34	85.79	79.55	87.50
	nbest	81.40	84.95	76.09	87.50
	strata	82.35	85.37	77.78	87.50
yes	none	92.10	89.29	97.22	87.50
	group	**93.51**	**91.37**	97.30	**90.00**
	nbest	92.10	89.29	97.22	87.50
	strata	92.31	90.91	94.74	**90.00**

Table 7.6: Relevance classification scores on the held-out testset, optimized towards F_2 (HO = hyperparameter optimization, FS = feature selection).

7.2.3 Selected hyperparameters and features

In this section, we take a closer look at the hyperparameters and features that are selected most often, and the differences between the various feature selection methods. We do this by studying the k-nearest fit solutions of each optimization run, i.e. all solutions that attain one of the top three fitness scores (see page 106).

FS	**none**	**group**	**nbest**	**strata**
Objective		**F_1**		
kernel	polynomial sigmoid	polynomial sigmoid	polynomial	polynomial
C (2^n)	-2, 4, 6, 10 4, 6	4, 6, 12 6	-4	-6, 6, 10
d	2, 3 n/a	2 n/a	3	2
γ (2^n)	-14, -12, -8, 0 -14	-4, -2 -14	4	-4, -2
Objective		**F_2**		
kernel	linear sigmoid	sigmoid	sigmoid	sigmoid
C (2^n)	4 2, 4	6	4	6
d	n/a n/a	n/a	n/a	n/a
γ (2^n)	n/a -14, -12	-14	-12	-14

Table 7.7: Selected hyperparameters for the relevance task.

Table 7.7 gives an overview of the selected hyperparameters after each run. It appears that the optimization objective affects kernel choice: when optimizing towards F_1, polynomial kernels are usually selected, whereas sigmoid kernels are used when optimizing towards F_2. Given that linear kernels are virtually lacking from these results would indicate that the relevance problem cannot easily be separated linearly. This also explains why hyperparameter optimization is so effective.

The results indicate that for polynomial kernels, the choice of C has little influence, and high degrees of freedom d are avoided. For the sigmoid kernels, C is limited to the 2^2 to 2^6 range, and the value for γ is very small.

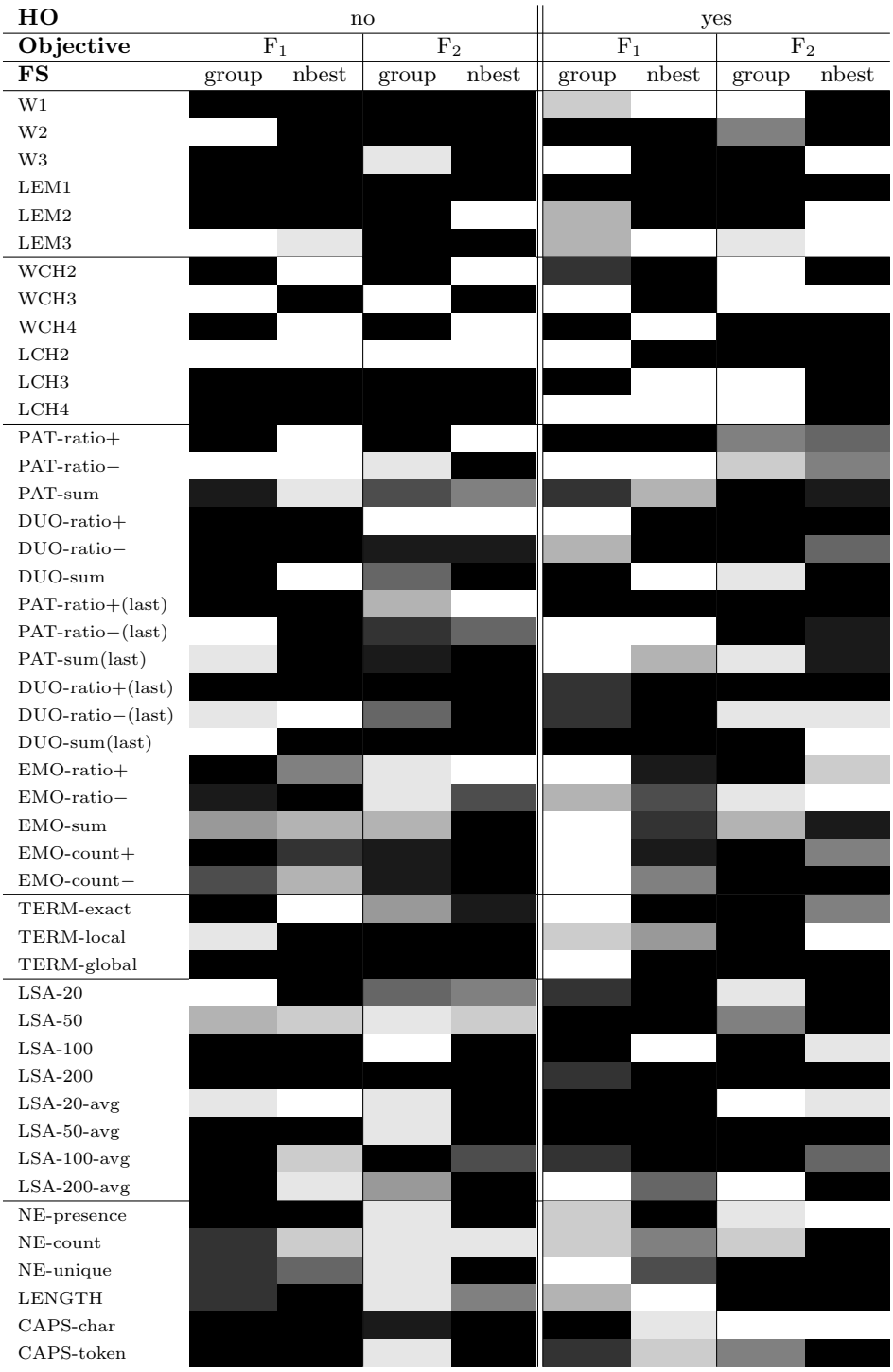

Table 7.8: Feature group selection status in all relevance models with regular or nbest feature group selection (FS), with or without hyperparameter optimization (HO), and optimized towards F_1 or F_2. Cell colour indicates the relative frequency of selection (darker = more often selected).

Table 7.8 summarizes the results of regular and nbest feature group selection; strata feature group selection is discussed separately below. Each column represents the results of one optimization experiment. In each cell, we code the selection status of a feature group with a grey value: the darker the tone, the more the feature is selected in the n-nearest fit solutions. For the W1 feature group in the top row, for example, we find that it is always selected (true black) in models without hyperparameter optimization. When it does not appear in any of the top solutions of a run, the cell is white. In all other cases, the grey value is proportionate to the ratio of solutions that select the feature group.

As mentioned previously, genetic algorithms do not cover the entire search space, and some allele choices in the solution set may not be optimal. This is why we average over a number of solutions to make the analysis more robust. A horizontal row in the table that is mostly dark would indicate a feature group that enjoys high selection status regardless of optimization strategy. We can then infer that it is highly informative. Likewise, dark regions indicate families of feature groups (e.g. token ngram features) that are successful with a specific optimization objective, or in combination with or without hyperparameter optimization. Finally, dashed rows (where dark and light cells alternate, e.g. WCH3 and WCH4) indicate that a feature group works better with regular than with nbest feature group selection, or vice versa.

First, we look at the best information sources for the relevance task. The following features are almost always selected:

- LEM1, W2 and LCH3 as bag-of-words information. Short token ngrams are preferred. This would indicate that the model relies on specific keywords, rather than longer collocations, to identify relevant posts.

- PAT-ratio+(last), DUO-ratio+(last), DUO-sum(last), DUO-ratio− and EMO-count+ from the lexicons. This list includes features from every lexicon. It appears that polarity information helps in distinguishing relevant from irrelevant posts. This tentatively confirms our prior assumption that negative (or lack of positive) polarity is associated with posts about suicide. Additionally, we find that it is helpful to calculate the features on the last tokens only.

- The TERM-global feature. These features are more informative than the token bigrams and trigrams. This validates the approach of extracting terms from a specialized corpus, in order to obtain highly salient collocations. The relaxed matching technique also provides better abstraction than the ngram or TERM-exact features.

- LSA-200, LSA-50-avg and LSA-100-avg, derived from the topic models. When

hyperparameter optimization is applied, LSA features seem to be especially useful.

None of the character ngram feature groups is unanimously selected. However, the results suggest that character sequences of words and lemmas are mutually interchangeable. Taking that into account, it is likely that the WCH2, LCH3 and LCH4 features would be selected even more if their counterparts were not available.

The miscellaneous feature groups do not appear to be particularly informative, especially not when hyperparameter optimization is applied. Overall, we can observe that the amount of selected features declines with hyperparameter optimization.

Table 7.9 outlines the selection results of the best classifier (joint optimization towards F_2 with stratified feature groups). On each line, the selection status of all bins for a feature group are presented. The bins are sorted according to informativeness, as determined with the information gain metric. The most informative bin is presented on the left. Dashes indicate the absence of a bin at a certain rank. With all the lexicon-based features, for example, only one 'bin' (containing the only feature) can be displayed.

The observations based on the regular and nbest feature group selection also hold for stratified selection: with the exception of DUO-ratio– and LEM1, the same feature groups are strongly selected. The large W2 and LSA-200 groups are entirely relevant, with virtually every bin selected.

The results can also serve as a validation for the information gain filtering metric. The bins to the left are supposed to be the most informative, so we should expect them to be selected more often. This is true for a number of feature groups: for W1, W3, WCH2, WCH4, LCH4 and LSA-100, the 'white tail' indicates that the least informative features are omitted. LEM2 and LCH3 are notable exceptions. There, the leftmost bins are not selected, as opposed to bins further to the right. However, this may be another indication that the original and lemmatized bag-of-words features are mutually redundant.

Lastly, the results demonstrate the potential benefit of using stratified feature selection. In the ngram feature groups, for example, more than half of the bins is removed. Not only did this result in better scores, it also makes for a model with a minimum of irrelevant features. Feature reduction amounts to nearly 60% (from 21 791 to under 10 000), similar to what is achieved with regular feature selection. The main difference is, however, that instead of having to include or exclude an entire feature group (as with regular and nbest selection), the search algorithm can pick the most useful subsets. This is indeed what

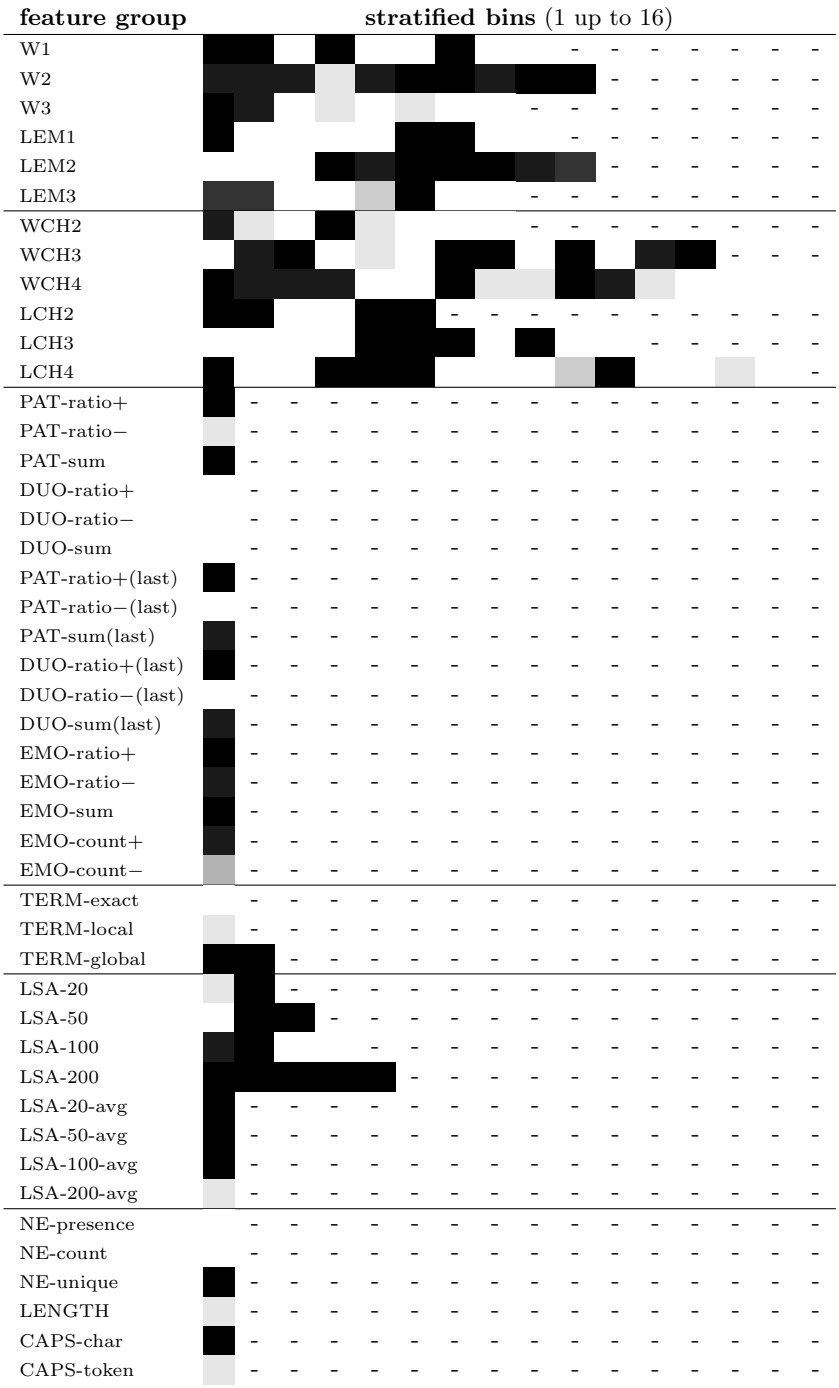

Table 7.9: Selection status of all feature group strata, in the relevance model with joint optimization towards F_2. Cell colour indicates the relative frequency of selection (darker = more often selected).

happens: we can observe that bins are selected from all feature groups.

7.3 Direct severity classification

In the previous section, we investigated whether a classification-based approach is viable for the detection of suicide-related posts in social media. The obtained models show promising results, especially after optimization. In this section, we look into the task of finding severe suicide-related posts to determine if it, too, can be successfully modeled, which features are informative, and whether model optimization can help in improving performance.

7.3.1 Cross-validation results

The results of the two sets of optimization experiments are listed in Tables 7.10 and 7.11, and displayed in Figure 7.4.

The obtained scores are markedly lower than for the relevance task, indicating that severity is harder to model. This is not entirely unexpected: differentiating between severe and non-severe messages assumes a prior understanding of which messages are suicide-related, and the severity distinction is more ambiguous. Examples 24 and 25, for example, are excerpts from posts that are both relevant to suicide, the latter of which is severe. Automatically determining the presence of a severe suicide risk is not trivial, considering the lexical overlap in the messages' content.

(24) NL: Echt waar joenge als ik van jou afscheid moet nemen, pleeg ik zelfmoord

EN: Seriously man if I need to leave you, I'll commit suicide

(25) NL: kwillet weg moet ik zelfmoord plegen om het uit mijn hoofd te krijgen?

EN: I want it gone do I have to commit suicide to get it out of my head?

In short, severity detection makes for a finer-grained problem. What is more, the number of training examples is much smaller, at less than one third the amount available for the relevance task.

Poor recall is the main obstacle. None of the models beat the baseline recall of 81.93%, although it should be noted that the baseline excels in this respect to

the detriment of its precision, which is very low at 23.24%. All models beat the baseline performance in terms of precision, F_1 and F_2.

When the results with F_1 and F_2 optimization are compared, we find that the F_2 objective significantly improves recall with nbest and stratified feature group selection, but fails to do so in combination with regular feature group selection. This may be because regular feature group selection is more effective at improving precision: it reaches a higher precision than the other selection techniques in three out of four comparisons.

Contrary to the relevance task, optimizing hyperparameters does not produce better models for severity. When no feature selection is applied, hyperparameter optimization fails to improve the results of the unoptimized model, so it brings no error reduction (see Figure 7.4). This indicates that the default settings cannot be improved upon, or rather, that models for severity are unaffected by different hyperparameters. Inspection of the set of elite individuals after optimization confirms this: a wide variety of solutions reach the same F_1 fitness score of 61.36%.

We can observe positive and negative trends in error reduction, when comparing feature selection techniques with or without hyperparameter optimization. When doing joint optimization, the error reduction is greater with nbest feature selection, but smaller with regular and F_2-optimized stratified feature group selection. These differences might be caused by the tuned hyperparameters, but given the absence of impact on the full feature set, we have reason to doubt this.

It is unusual that some results obtained after joint optimization are not on a par with those where only feature selection was performed, if hyperparameters indeed have no impact. We see two explanations for these variations. Firstly, the larger search space that comes with joint optimization makes the optimization harder. With identical GA settings, an optimization run on a smaller search problem is likely to yield better results. Secondly, when a series of seemingly irrelevant hyperparameters is added to the search problem, we expand the genome with genes that have no impact on fitness. When a new population is created, all individuals where crossover or mutation occurred in this portion of the genome will achieve the same fitness as their parents. A population is therefore less likely to produce new elite individuals. As a consequence, the search algorithm is more likely to terminate prematurely, when the elite fitness score has not changed for five generations in a row.

HO	FS	F_1	F_2	precision	recall
baseline		36.21	54.44	23.24	**81.93**
no	none	61.36	57.40	69.31	55.04
	group	**69.04**	63.93	79.67	60.92
	nbest	67.13	62.99	75.39	60.50
	strata	67.29	62.77	76.47	60.08
yes	none	61.36	57.40	69.31	55.04
	group	68.88	63.87	79.23	60.92
	nbest	68.54	**64.03**	77.66	61.34
	strata	67.33	60.25	**83.75**	56.30

Table 7.10: Direct severity classification scores on the development set, optimized towards F_1 (HO = hyperparameter optimization, FS = feature selection).

HO	FS	F_1	F_2	precision	recall
baseline		36.21	54.44	23.24	**81.93**
no	none	61.36	57.40	69.31	55.04
	group	66.82	62.88	74.61	60.50
	nbest	65.95	64.94	67.70	64.29
	strata	**69.51**	**66.81**	74.52	65.13
yes	none	61.36	57.40	69.31	55.04
	group	67.29	62.50	**77.17**	59.66
	nbest	68.92	66.07	74.27	64.29
	strata	66.96	64.81	70.89	63.45

Table 7.11: Direct severity classification scores on the development set, optimized towards F_2 (HO = hyperparameter optimization, FS = feature selection).

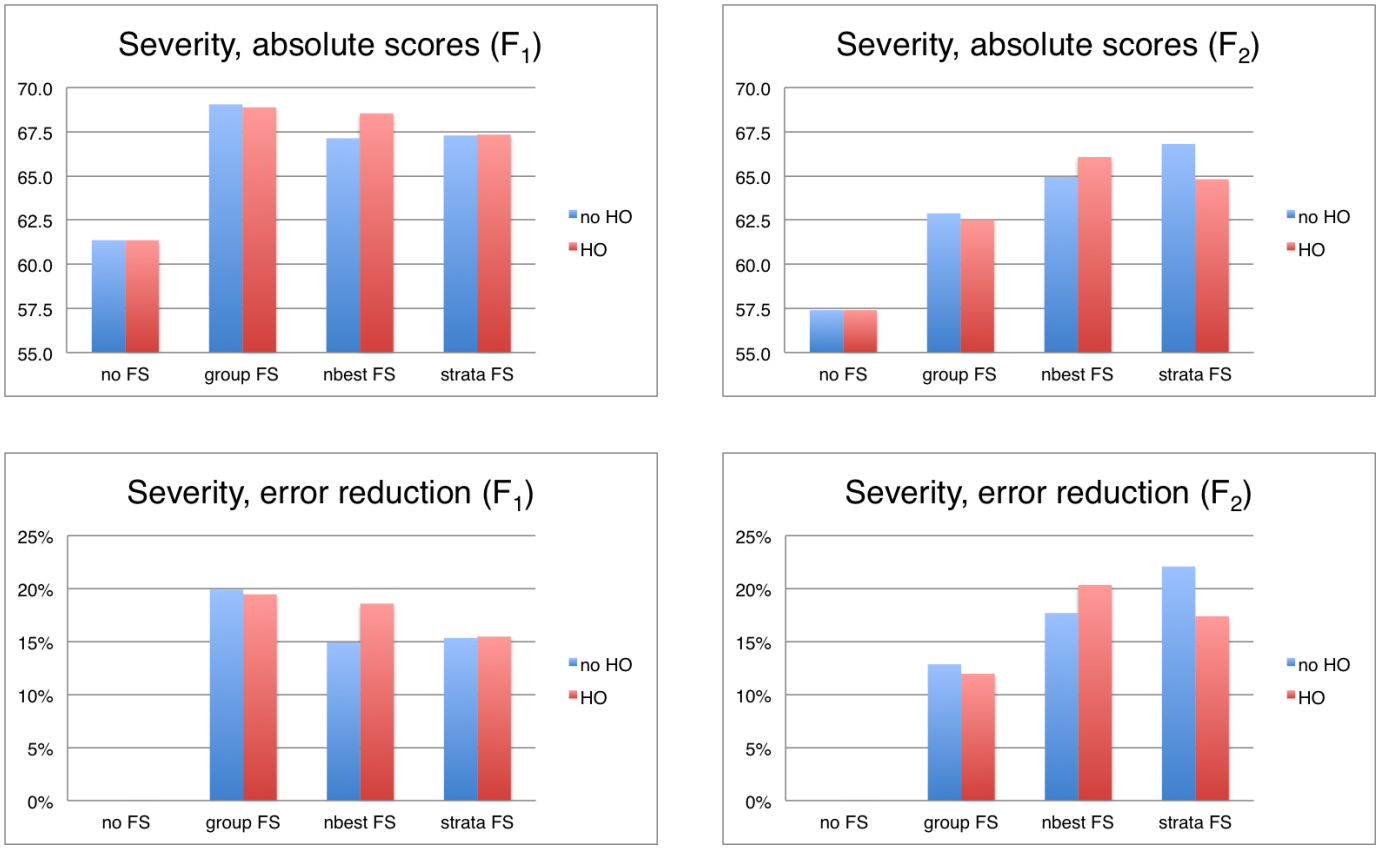

Figure 7.4: Absolute scores (top row) and percentage-wise error reduction after optimization (bottom row), for the severity task on the development set. Scores are expressed in terms of F_1 (left) or F_2 (right), depending on the optimization objective.

In absolute scores, a performance gain of more than 8 percentage points is achieved with optimization, by virtue of feature selection. The absolute gains for the relevance task were smaller at 2 percentage points, but optimization had to start from much higher unoptimized scores. In terms of error reduction, the effect is comparable at around 20% reduction.

The best solution achieves a F_1 score of 69.51%, with 74.52% precision and 65.13% recall. It is obtained with stratified feature selection without hyper-parameter optimization, and as with the relevance experiments, it is produced after F_2 optimization. Optimizations with regular feature group selection also produce good F_1 results, mainly because of high precision. Approaches with nbest and stratified feature selection achieve the highest recall and F_2 scores.

7.3.2 Results on the held-out testset

When training the optimized models on all development data, and testing them on the held-out testset, we obtain the results listed in Tables 7.12 and 7.13.

Again, the high baseline scores on the testset are striking, and warrant some clarification. The high recall of 80% is caused because the baseline does not distinguish between relevant and severe posts: all posts containing a keyword are considered relevant and severe. This is a strategy that causes low precision, but given the 50/50 balance of severe and non-severe posts in the held-out set, precision is acceptable at 47.06%. As a consequence, F_1 and F_2 are also high. The baseline scores on the large development dataset give a more realistic image, with precision at 23.24%, and the according lower F-scores.

On the development dataset, we noticed that all models had difficulty achieving good recall, and were optimized more towards precision. The same is true on the testset, with models finding between 7 and 11 of the 20 severe posts. Surprisingly, the models optimized towards F_2, which achieved better recall on the development set than the F_1-optimized models, perform worse on the testset. The F_1-optimized models do outperform their F_2 counterparts in terms of precision.

The models that were optimized with regular feature group selection achieved the best precision on the development data, and the same observation can be made on the test set, with precision scores around 80%. Combined with comparatively good recall, these models attain the highest F_1 and F_2 scores (66.67% and 59.14%, respectively).

Overall, the observations made on the development set largely hold on the test set as well. Although it is clear that the severity task is harder than the relevance

HO	FS	F_1	F_2	precision	recall
baseline		59.26	**70.18**	47.06	**80.00**
no	none	54.05	51.55	58.82	50.00
	group	**66.67**	59.14	**84.62**	55.00
	nbest	45.00	45.00	45.00	45.00
	strata	53.33	57.14	48.00	60.00
yes	none	54.05	51.55	58.82	50.00
	group	**66.67**	59.14	**84.62**	55.00
	nbest	46.15	45.45	47.37	45.00
	strata	56.25	48.91	75.00	45.00

Table 7.12: Direct severity classification scores on the held-out testset, optimized towards F_1 (HO = hyperparameter optimization, FS = feature selection).

HO	FS	F_1	F_2	precision	recall
baseline		**59.26**	**70.18**	47.06	**80.00**
no	none	54.05	51.55	58.82	50.00
	group	50.00	43.48	66.67	40.00
	nbest	41.03	40.40	42.11	40.00
	strata	43.24	41.24	47.06	40.00
yes	none	54.05	51.55	58.82	50.00
	group	53.33	44.44	**80.00**	40.00
	nbest	40.00	36.84	46.67	35.00
	strata	42.86	44.12	40.91	45.00

Table 7.13: Direct severity classification scores on the held-out testset, optimized towards F_2 (HO = hyperparameter optimization, FS = feature selection).

task, we can state that the obtained results would be usable in practice. The models achieve high precision, and are clearly more informed than a keyword-based baseline, which cannot distinguish between relevant and severe posts. However, better recall would be desirable. We therefore investigate the effect of cascaded severity classification in Section 7.4. Cascaded classification may allow a severity classifier to sacrifice more precision for better recall, since the set of potential false positives is reduced by the relevance classifier that came before it.

7.3.3 Selected hyperparameters and features

For completeness, we list the results of the hyperparameter optimization in Table 7.14. As discussed above, we found that different hyperparameters did not seem to have an impact on model performance, so we will not discuss them in detail. The hyperparameters tested without feature selection covered the majority of the search space, so for brevity, they are omitted from the table. When feature selection was enabled, the search tended towards simplicity, with linear models.

FS	**group**	**nbest**	**strata**
Objective		**F_1**	
kernel	linear	linear	linear polynomial
C (2^n)	2, 10, 12	-4	-4 -6, 8
d	n/a	n/a	n/a 2
γ (2^n)	n/a	n/a	n/a -2, 0, 2
Objective		**F_2**	
kernel	linear	linear	linear
C (2^n)	-2, 10, 12	4, 8	4
d	n/a	n/a	n/a
γ (2^n)	n/a	n/a	n/a

Table 7.14: Selected hyperparameters for the direct severity task.

Table 7.15 lists the feature groups selected with regular and nbest selection. The results for severity differ in a number of ways from those for relevance. A first observation is that the amount of selected features is lower, indicating that in general, fewer features are informative for the severity task.

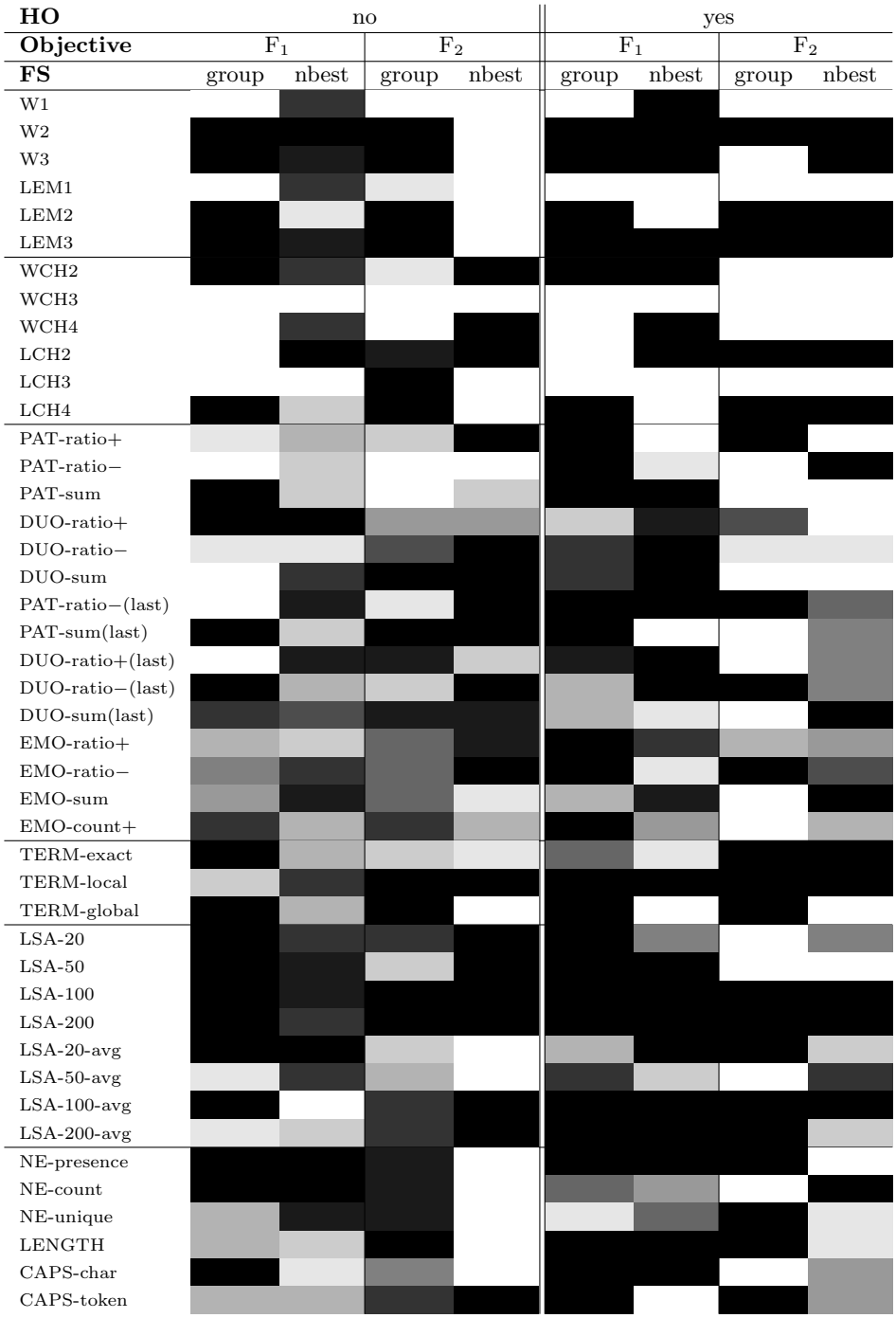

Table 7.15: Feature group selection status in all severity models with regular or nbest feature group selection (FS), with or without hyperparameter optimization (HO), and optimized towards F_1 or F_2. Cell colour indicates the relative frequency of selection (darker = more often selected).

We find the following feature groups to be consistently selected:

- W2, W3 and LEM3 in the token ngram features. The unigram features (W1 and LEM1), on the other hand, are discarded. For relevance, we observed an opposite trend, with shorter ngrams being preferred. It would seem, then, that for severity, the added specificity of collocations is useful, and simple keywords are not informative. The trigram *never commit suicide*, for example, would be much more informative for determining severity than its constituent unigrams *never*, *commit* and *suicide* in isolation, whereas for relevance detection, the unigrams would probably suffice.

- With character ngram features, we observe the same trend as for relevance: word and lemma character sequences are interchangeable, and most solutions choose one of the two alternatives. Ignoring the difference between words and lemmas, we find that character ngrams of length two and four are always selected, ngrams of length three only once.

- Contrary to the results for relevance, none of the polarity features are selected consistently. PAT-ratio−(last), PAT-sum(last) DUO-ratio−(last) and EMO-ratio− are often selected, suggesting that negative polarity is correlated with severe posts. Polarity features are more informative when calculated on the last tokens of a message, rather than the entire message.

- TERM features with non-exact matching are always included. As for the relevance task, the relaxed matching strategies are more robust than exact matching.

- The topic model features appear to be very informative, especially when the number of topics is high: the LSA-100 and LSA-200 are included in every model. This makes intuitive sense: the topics are generalizations of semantically related words, some of which may signal aspects of suicidality. Finer-grained topics could be more adequate to model such aspects.

- From the miscellaneous features, only named entity information is selected often. We speculate that these features may help in labeling informative and journalistic messages as non-severe.

The features selected by the best model on the development data (stratified feature selection without hyperparameter optimization, F_2 objective) are outlined in Table 7.16, for reference. The feature preferences do not diverge notably from the ones discussed above, although we can still observe internal selection in the informative feature groups. As with the relevance experiments, we find that feature bins in the 'tail' of a group are not necessarily discarded.

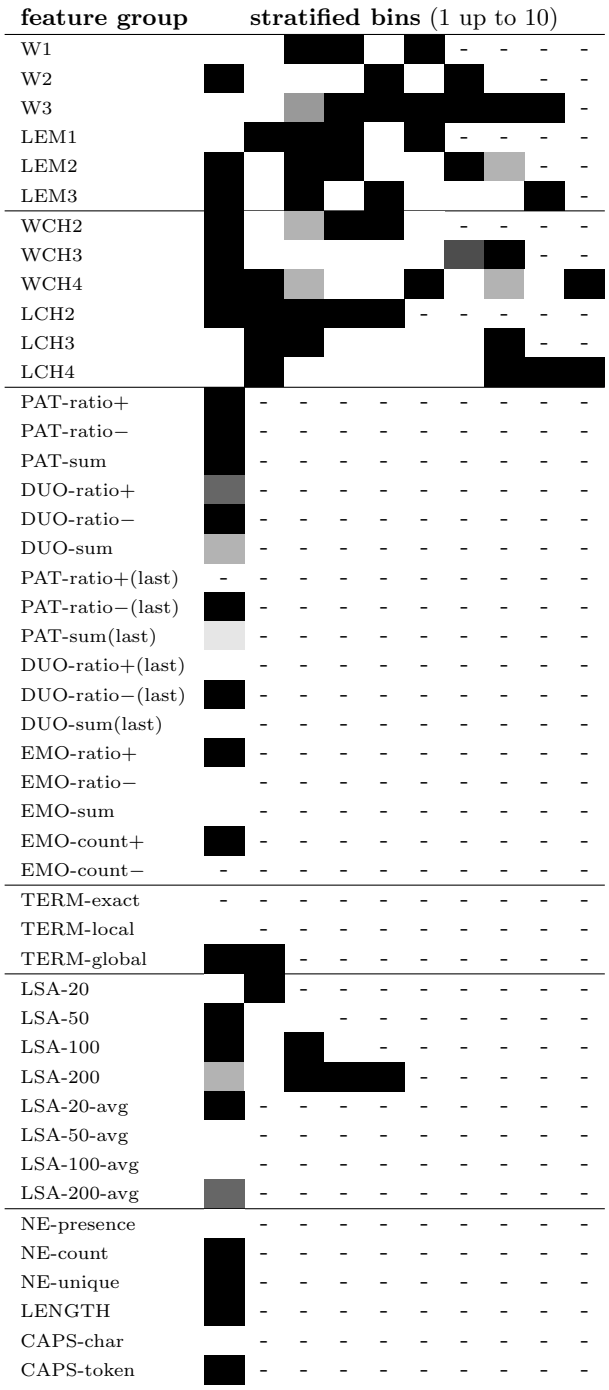

Table 7.16: Selection status of all feature group strata, in the severity model without hyperparameter optimization, optimized towards F_2. Cell colour indicates the relative frequency of selection (darker = more often selected).

7.4 Cascaded severity classification

In the previous section, we tested a direct classification approach for the severity task. We found that high precision can be attained, but recall is low. In this section, we investigate whether cascaded classification is a better approach for the severity task, and whether it can be used to remedy the recall problem. More specifically, we use a relevance classifier to filter the instances first, and then do severity classification on the instances classified as relevant.

We introduced the potential benefits of cascaded classification in Chapter 5. In summary, it can be used to take advantage of the decision tree structure of the annotation scheme. When a relevance classifier is used to make predictions at an intermediate level in the tree, a downstream severity classifier only has to model a branch of the tree. The severity model can therefore better fit the subproblem, and have lower bias.

In a first set of experiments, we test cascaded severity classifiers on the development set using gold standard relevance labels to filter the instances. This allows us to find the ceiling performance for cascaded classification, since there is no error percolation from the previous step. Next, we test cascades where filtering is done with a relevance classifier from Section 7.2, contrast their results with the ceiling performance, and test them on the held-out data.

The reduced number of instances allows us to test both LIBSVM and TiMBL for severity classification. To reduce experimental complexity, the Gallop optimization runs are all done with F_1, our primary performance metric, as the fitness objective.

7.4.1 Results with gold standard relevance labels

When gold standard relevance labels are used to filter the instances, the number of instances in the development set is reduced to 811. The optimization experiments for cascaded severity classification were run on this reduced dataset.

Table 7.17 presents the results. We include the baseline scores, and the best scores for direct severity classification on the development data, from the F_2-optimized SVM classifier with stratified feature selection and unoptimized hyperparameters.

Unlike with direct classification, the LIBSVM results reveal some positive impact from hyperparameter optimization, evidenced by the 1.39% F_1 increase over the unoptimized model without feature selection. This potential for better results

	HO	FS	F_1	F_2	precision	recall
baseline			36.21	54.44	23.24	81.93
best direct	no	strata	69.51	66.81	74.52	65.13
LIBSVM	no	none	69.20	66.70	73.81	65.13
		group	74.11	71.43	79.05	69.75
		nbest	72.10	71.19	73.68	70.59
		strata	**78.02**	**76.83**	80.09	**76.05**
	yes	none	70.59	67.48	76.47	65.55
		group	73.78	71.31	78.30	69.75
		nbest	70.90	69.17	73.97	68.07
		strata	76.06	73.21	**81.34**	71.43
TiMBL	no	none	62.47	62.56	62.34	62.61
		group	66.81	67.56	65.59	68.07
		nbest	63.73	67.21	58.66	69.75
		strata	69.73	69.99	**69.29**	70.17
	yes	none	68.00	73.97	59.94	78.57
		group	71.50	78.96	61.77	84.87
		nbest	71.31	**80.59**	59.83	**88.24**
		strata	**71.83**	76.43	65.29	79.83

Table 7.17: Cascaded severity classification scores on the development set, with gold standard relevance labels.

is not accomplished when hyperparameter optimization is combined with the three feature selection techniques. For each technique, we should expect the results after joint optimization to at least match those without hyperparameter optimization. However, they are consistently lower. We therefore revisit our earlier hypothesis (page 120) that this is caused by the expansion of the search space.

In an additional experiment, we tested whether this problem can indeed be remedied by altering the genetic algorithm settings. We reran the joint optimization with stratified feature selection, but increased the population size from 100 to 300. Optimization converged after 37 generations, with a best F_1 score of 78.05%. This result is an improvement over the run with a smaller population (76.06%, 31 generations), and it matches the score obtained without hyperparameter optimization (78.02%, 39 generations). We can conclude that the lower scores with joint optimization are indeed caused by the GA settings.

As is to be expected with gold standard relevance labels, the cascaded LIBSVM models compare favorably to the best results obtained with direct classification: all optimized models achieve higher recall, F_1 and F_2, and only the unoptimized model performs worse. Stratified feature selection yields the best F_1 results at 78%, which marks an 8.5 percentage point improvement (28% error reduction) over direct classification. As with direct classification, SVM produces high-precision models with lower recall, and as a result, F_2 always trails F_1.

The opposite is true for TiMBL: it is less conservative in its predictions, which promotes recall and lowers precision. Hyperparameter optimization has a positive impact on performance. Without it, most models fail to beat the F_1 score of the direct severity classifier. The improvements are accomplished through better recall. As a result, TiMBL is the better classifier in terms of F_2, scoring 80.59%. The best F_1 score for TiMBL, at 71.83%, is obtained when stratified feature group selection is combined with hyperparameter optimization.

Table 7.18 lists the selected hyperparameters. For LIBSVM, linear and polynomial kernels are preferred. For TiMBL, *overlap* as a distance metric is not selected, in favour of the *cosine* and *dot product* metrics. There is no clear preference for a feature weighting metric, possibly because *information gain* had already been used to remove the least informative features. The best results are obtained with a large neighbourhood, with a k-value of 9 being most frequently selected. To infer the class from the nearest neighbours, the more informed *inverse distance* and *inverse linear* voting strategies are preferred over normal majority voting.

Feature-wise, Table 7.19 shows preferences similar to those for the direct severity classification approach, with the TERM and LSA features being the most

FS	**none**	**group**	**nbest**	**strata**
Algorithm		**LIBSVM**		
kernel	linear	linear		linear
	polynomial		polynomial	
C (2^n)	0	6		10
	-6 to 12		0, 6, 8	
d	n/a	n/a		n/a
	2, 3		2	
γ (2^n)	n/a	n/a		n/a
	-10 to 4		-6, -2, 0	
Algorithm		**TiMBL**		
distance metric	C, D	C, D	D	C
feature weighting	None, IG, χ^2	GR, χ^2	GR	χ^2
k-value	5, 7, 9	7, 9	7, 9	9
voting	Z, ID, IL	ID, IL	Z, ID	IL

Table 7.18: Selected hyperparameters for the cascaded severity task with gold standard relevance labels.

important. Cascaded classifiers, which need not differentiate between relevant and irrelevant posts, depend even less on token ngram features, while the WCH2 and WCH4 character ngrams are still often selected. Lexicon information does not seem to be essential, although again, the features about emoticons (EMO) and those calculated on the final words (last) are popular.

To summarize, these experiments have shown the maximum performance that can be achieved with cascaded classification, given a perfect relevance filter. We find that the severity task, in isolation, can be modeled with success, and that LIBSVM optimizes towards precision, TiMBL towards recall. The question remains, however, whether the performance gains compared to direct classification are caused by more than a lack of error percolation from the previous step. This is investigated in the next section.

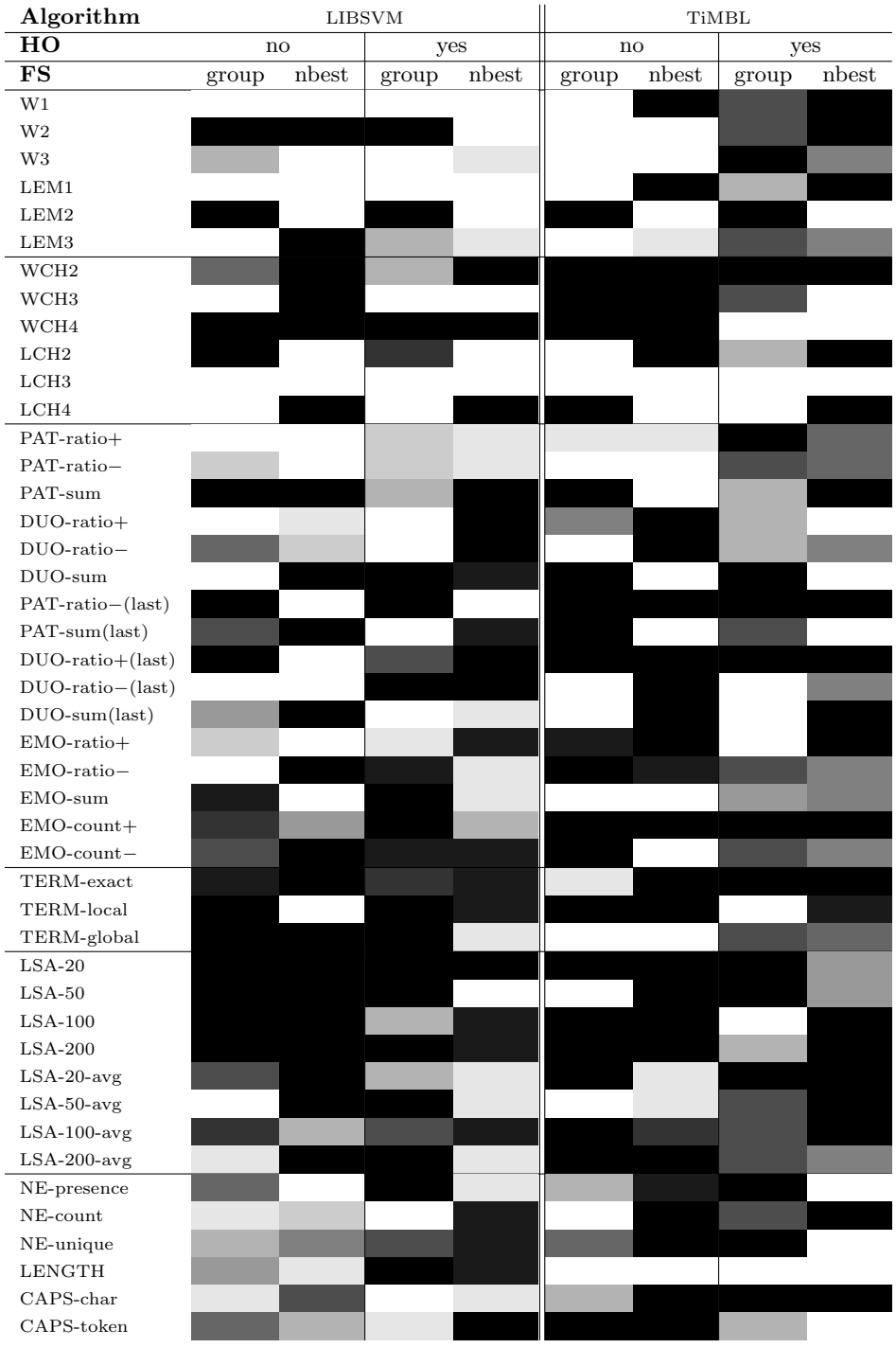

Table 7.19: Feature group selection status in cascaded severity models with regular or nbest feature group selection (FS), with or without hyperparameter optimization (HO), and using gold standard relevance labels. Cell colour indicates the relative frequency of selection (darker = more often selected).

7.4.2 Results with predicted relevance labels

In Section 7.2, we obtained sixteen optimized classifiers for relevance, which we can use as the first step in a cascade. Recall is an important factor in the choice of relevance filter, since false positives can be corrected by a downstream classifier, whereas false negatives cannot. Not all false negatives are problematic, however: non-severe relevant posts that are missed will not affect severity recall.

We experiment with the relevance filters that achieved the highest F_1, F_2 and recall scores. All of these were obtained with hyperparameter optimization: the F_1-optimized model with stratified feature selection, and the F_2-optimized models with stratified and nbest feature selection. We will refer to them as the F_1-strata, F_2-strata and F_2-nbest relevance filters. Using these filters, the dataset is reduced to 813, 830 and 844 instances, respectively.

These filters are combined with LIBSVM and TiMBL cascaded severity classifiers, optimized towards F_1. We do the optimization with and without hyperparameter tuning, and use stratified feature group selection, the overall best technique in the gold standard experiments. Results on the development set are listed in Table 7.20, and on the held-out testset in Table 7.21.

The LIBSVM results with predicted relevance labels are significantly lower than with gold standard relevance filtering. This is mainly caused by a dramatic drop in recall of around 15%. This indicates that the effect of error percolation from the relevance step is substantial.

Compared to the best direct classifier, LIBSVM cascades perform worse in terms of recall, F_1 and F_2. Although precision is generally better than that of the best direct classifier, it is below the best precision of some other direct classifiers. Cascades with the F_2-strata filter achieve the best scores for F_1 at 68.86% and F_2 at 64.36%, which is 0.65% and 2.45% below the performance of the best direct classifier. The same models perform well on the held-out data, also scoring below the best results obtained with direct classification.

With TiMBL, the results with predicted relevance labels are much closer to those with gold standard filtering. TiMBL's tendency towards strong recall performance remains. Three out of six models achieve better recall and F_2 than the best direct classifier, which was the best direct classifier in terms of both F_1 and F_2. The F_1-strata model without hyperparameter optimization does this by overgenerating positive predictions: it achieves high recall but very low precision. The F_1-strata and F_2-nbest models with hyperparameter optimization, on the other hand, achieve a better balance, and therefore also score well in terms of F_1. None of the TiMBL models improve the best F_1 score obtained with direct classification, however. On the held-out data, we notice a drop in

	relevance filter	HO	FS	F_1	F_2	precision	recall
baseline				36.21	54.44	23.24	81.93
best direct		no	strata	69.51	66.81	74.52	65.13
LIBSVM	gold standard (811)	no	strata	78.02	76.83	80.09	76.05
	F_1-strata (813)	no	strata	65.39	60.46	75.69	57.56
		yes	strata	67.43	63.64	74.87	61.34
	F_2-strata (830)	no	strata	**68.86**	64.14	**78.49**	61.34
		yes	strata	68.69	**64.36**	77.37	**61.76**
	F_2-nbest (844)	no	strata	68.21	64.19	76.17	**61.76**
		yes	strata	63.79	58.80	74.30	55.88
TiMBL	gold standard (811)	yes	strata	71.83	76.43	65.29	79.83
	F_1-strata (813)	no	strata	63.21	**72.03**	52.50	**79.41**
		yes	strata	**67.78**	67.96	67.50	68.07
	F_2-strata (830)	no	strata	64.15	64.24	64.02	64.29
		yes	strata	63.92	60.76	**70.00**	58.82
	F_2-nbest (844)	no	strata	63.38	64.17	62.10	64.71
		yes	strata	65.47	67.49	62.36	68.91

Table 7.20: Cascaded severity classification scores on the development set, with automatically predicted relevance labels.

	relevance filter	HO	FS	F_1	F_2	precision	recall
baseline				59.26	70.18	47.06	80.00
best direct		no	group	66.67	59.14	84.62	55.00
LIBSVM	F_1-strata (37)	no	strata	45.16	38.46	63.64	35.00
		yes	strata	42.86	34.09	**75.00**	30.00
	F_2-strata (38)	no	strata	57.14	52.63	66.67	**50.00**
		yes	strata	**58.82**	**53.19**	71.43	**50.00**
	F_2-nbest (36)	no	strata	40.00	33.33	60.00	30.00
		yes	strata	54.55	48.39	69.23	45.00
TiMBL	F_1-strata (37)	no	strata	**68.29**	**69.31**	66.67	**70.00**
		yes	strata	60.61	53.76	**76.92**	50.00
	F_2-strata (38)	no	strata	48.49	43.01	61.54	40.00
		yes	strata	55.56	52.08	62.50	50.00
	F_2-nbest (36)	no	strata	47.06	42.55	57.14	40.00
		yes	strata	51.43	47.37	60.00	45.00

Table 7.21: Cascaded severity classification scores on the held-out testset, with automatically predicted relevance labels.

recall for most models, with the exception of the first. Not unlike the baseline, its overgeneration strategy gives good recall and acceptable precision, allowing it to reach the best F_1 and F_2 scores on this data set.

In conclusion, we find no evidence that cascaded classification is more effective than direct classification, when the aim is to have a model with good balance between precision and recall. We expected cascaded classification to reduce confusion on the severity task, but this is not reflected in higher precision. LIBSVM models do achieve high precision, but this is offset by a loss in recall, caused in part by errors at the relevance classification stage.

If recall is considered to be more important than precision, and the according aim is high F_2, TiMBL models offer an advantage. However, we cannot claim that this is a consequence of cascaded classification, since we have no direct classification results from TiMBL to compare with. One particular advantage of cascades is that they allow the usage of more resource-intensive components like TiMBL, where they might not be usable in a direct classification setup for technical (e.g. memory requirements) or practical reasons (e.g. limited speed).

7.5 Scaling and error analysis

So far, we have reported results on datasets with a high incidence of suicide-related material. In this section, we test our models on datasets of increasing size (described in Section 7.1.1), to approximate the class skew and scale of 'big data' found on social media platforms.

Since these scaling datasets have not been annotated, we do not know if they contain suicide-related posts. We therefore cannot report on the recall of our models. However, the scaling experiments can shed some light on their usability in terms of precision: as the size of our dataset increases, how many false positives (i.e. noise) are added to the small number of known true positives? Would the amount of noise make the system unpractical in a real-world situation, which would involve the ongoing evaluation of large quantities of posts?

For each task, we selected the two direct classifiers with the best F_1 and F_2-optimized performance on the development set. For relevance, the best models are the ones obtained with hyperparameter optimization combined with stratified and nbest feature group selection, for F_1 and F_2, respectively. The best severity classifiers were obtained without joint optimization, and with group (F_1) or stratified (F_2) feature group selection.

These models were used to detect relevant or severe posts in the held-out and

scaling datasets. The posts that had been predicted as positive were subjected to a qualitative analysis, in which we categorized them into three groups:

- **True positive** instances, i.e. posts which are indeed relevant or severe, depending on the task. The majority of these instances are part of the held-out dataset, although 3 relevant messages (of which 2 severe) were detected in the scaling datasets as well.
- False positive instances that contain **risk factors**. These are posts that would not have been classified as relevant or severe by human annotators, but that contain mentions of difficult situations, such as drug abuse, relationship problems, or the death of a loved one.
- False positive instances that are **irrelevant**. Unlike the previous category, these posts do not contain any content that could be considered a risk factor, and can therefore be considered as true noise.

Figure 7.5 shows the results of this analysis for each classifier.

7.5.1 Relevance

The results of the relevance classifiers show that the number of hits hardly goes up as dataset size increases. The amount of noise, therefore, is minimal even on the full 300 000 post corpus, with only two false positives from the best F_1 and four from the best F_2 classifier. Half of these can be considered truly irrelevant (Example 26), the others contain risk factors that explain why the post may have been selected, such as the hurt expressed in Example 27. Notably, the sets of false positives produced by either system do not overlap.

(26) NL: [...] Hierover is ook een grote onzekerheid, sommige mensen beweren zelfs dat hij nooit heeft bestaan en dat Homerus een verzamelnaam was voor de mensen die de Ilias en de Odyssee hebben opgeschreven.

EN: [...] There is great uncertainty about this, some people even claim that he never existed and that Homer was a collective name used by the authors of the Iliad and the Odyssey.

(27) NL: waarom hebben ze mijn zusje ontnomen. het leven is keihard en verdomd moeilijk. ik heb zo'n verdriet en pijn [...]

EN: why was my sister taken from me. life is tough and damn hard. I am so sad and hurt [...]

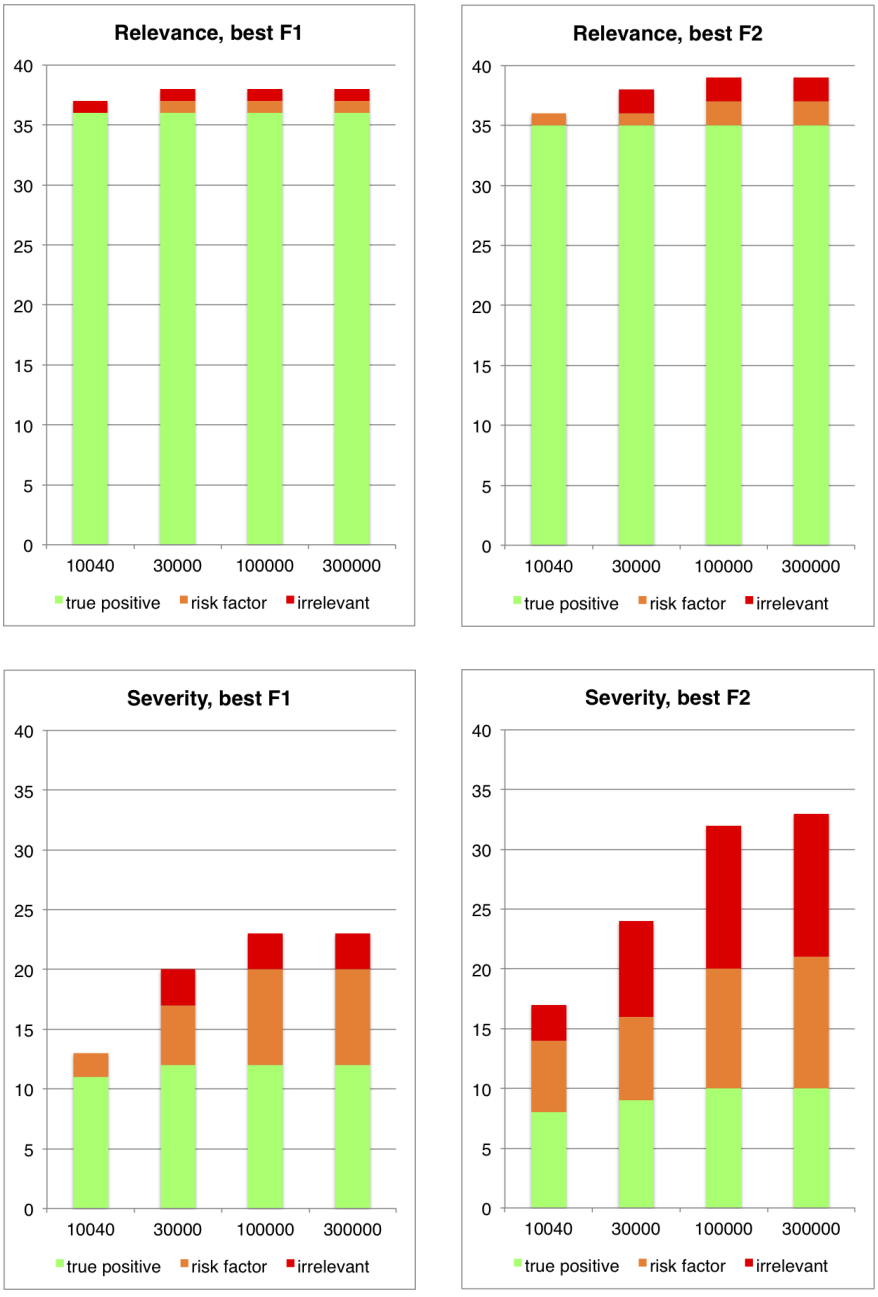

Figure 7.5: Number of true and false positives on the held-out and scaling datasets, using the best F_1 and F_2 classifiers for the relevance and severity tasks. False positives are divided into two groups, depending on whether they contain risk factors.

For an analysis of the false negatives, we can only look into the 40 posts in the held-out set that are known to be relevant. Four of these were missed by the F_1 classifier (two of which severe). The F_2 classifier missed the same four, and one additional severe post. The two non-severe posts both contain a fictitious account about suicide (Example 28). Contrary to the majority of the true positives in our sample, none of the false negatives contain explicit mentions of the words *zelfmoord* or *zelfdoding*. Instead, evocative imagery or vague wording is used to describe suicidal intent, such as in Examples 29 and 30, from a post found by the F_1 model, but missed by the F_2 model.

(28) NL: [...] TOEN PAS SCHOOT ZE IN PANIEK ZE NAM EEN MES EN... SMEER EEN BOTERHAM MET PINDAKAAS, ZE NAM EEN TOUW EN... HANGDE DE WAS OP MAAR NU GEBEURDDE HET ZE NAM EEN PISTOOL EN SCHOOT HAAR ZELF DOOD. [...]

EN: [...] ONLY THEN SHE PANICKED SHE TOOK A KNIFE AND... SPREAD A PEANUT BUTTER SANDWICH, SHE TOOK A ROPE AND... HUNG THE LAUNDRY BUT NOW IT HAPPENED SHE TOOK A GUN AND SHOT HERSELF. [...]

(29) NL: [...] mijn leven is op
het is klaar, mijn strijd [...]

EN: [...] my life is done
it is finished, my fight [...]

(30) NL: [...] mijn bloed donkerrood
samen met het water
laat ik me gaan

EN: [...] my blood crimson
along with the water
I let myself go

Although the relevance classifiers succeed in finding some of these messages, it is clear that it generally errs on the conservative side and mainly tags posts that will certainly be suicide-related. This is not entirely surprising: given the data collection approach that had to be taken to obtain positive material, the frequency of such 'explicit' posts is high in our data. The conservative strategy also explains the high precision observed in the scaling experiment.

7.5.2 Severity

The scaling results of the severity classifiers indicate that they make many positive predictions outside the held-out testset, unlike the relevance classifiers. This is especially true for the F_2 model optimized for higher recall. The amount of noise is still controlled, however: on the full dataset, the F_1 model produces about 50% false positives, most of which contain mentions of risk factors. Noise is higher in the F_2 model, with roughly one third true positives, one third false positives with risk factors and one third irrelevant false positives.

The severity models find four relevant posts that had been missed by the relevance models. Three of those are true positives, i.e. severe instances. The non-severe relevant post contains a reference to past suicidal ideation (Example 31). It appears that the severity models are better capable of detecting subtle references to suicide (Example 32), presumably by relying less on token unigram features to act as keywords, and more on longer collocations and LSA features.

(31) NL: ken da ke ook veel meegemaakt
me paardjes e me der door geholpen zonder under was ek ier nie mee [...]

EN: I know, I've also been through a lot
my horses have always helped me come through without them I wouldn't be here anymore

(32) NL: [...] zou het ni beter zijn als ik nu zou gaan?
ik wil worden verlost van myn pijn en verdriet! [...]

EN: [...] wouldn't it be better if I would go now?
I want to be freed from my pain and sorrow! [...]

Apart from the relevant ones, the F_1 model detects another five posts containing risk factors, the F_2 model finds nine. The risk factors are varied, and include relationship problems (e.g. Example 33), loneliness, the death of friends or pets, illness, bullying and anger.

(33) NL: [...] Maar een echte vriendin? Mijn hart is gebroken, alles is weg, ik had nu wat hoop, was gelukkig, maar dat is niet meer. [...]

EN: [...] But a real friend? My heart is broken, all is gone, I had some hope, was happy, but not anymore. [...]

There are also irrelevant false positives without risk factors. On the full dataset, the F_1 model produces three, the F_2 model twelve. About half of these (one

and seven, respectively) contain phrases that likely caused confusion, such as the love declaration in Example 34.

(34) NL: [...] Tot het einde bby, zelfs langer dan het leven ng samen verder in het Hiernamaals

EN: [...] Until the end bby, even longer than life together into the Hereafter

Recall on the held-out testset is low for severity, with nine known false negatives for the F_1 model. The F_2 model misses the same nine instances, and three others. Inspection of the errors reveals three potential causes. Two posts were also missed on the relevance level, likely for the same reason (vague wording). In another four, suicide is mentioned more explicitly, but it is unclear whether they are generic references, or apply to the author. For example, in 35, a story is told that may or may not be about the author. These are examples that would be ambiguous for human annotators as well. The third category of errors contains four posts in which the author mentions the potential suicide of a third person (e.g. Example 36). It should be noted that all posts that are about third persons that have committed suicide in the past are correctly labeled as non-severe. The models possibly discard all posts that are not specifically about the author.

(35) NL: [...] Door het verlies van de man van haar leven
besliste ze om zelfmoord te plegen
zij kon hem niet vergeten en hij...
Trok het hem niet aan en ging verder met zijn leven.

EN: [...] Because she lost the man of her life
she decided to commit suicide
she could not forget him and he...
Did not care and went on with his life.

(36) NL: wat moet je doen als iemand overal in zijn omgeving blijft dreigen me zelfmoord en zo mensen probeert te chanteren of terug voor zich te winnen. moet je dat au serieux nemen ? [...]

EN: what should you do when someone in his environment keeps threatening to commit suicide and tries to blackmail people or win them back. should you take it seriously ? [...]

We can conclude that the severity models are moderately successful in detecting posts in which an author personally discloses suicide ideation. When this is done in explicit terms, recall is high. Further research should be conducted to improve

performance on posts that contain vague references to suicide. Considering that the noisiest model predicts less than 30 false positives on a corpus of 300 000 posts (or less than 0.01%), noise is acceptably low for the system to be usable in a real-world application.

7.6 Summary

The main research objective of this study is to explore the automatic detection of relevant and severe user-generated content related to suicide, and determine which information sources and methodological approaches aid performance. To that end, a series of experiments was conducted, the results of which were presented and analysed in this chapter. We briefly list the main observations.

A classification-based approach to the detection of suicide-related material is viable. The relevance task can be carried out with high precision and recall, and the proposed system scales well to large datasets with high class skew. High-risk content is more complex to recognize, both for human annotators and machine learning models. Nevertheless, the obtained severity detection system can be considered a step forward in automated prevention practice. In future work, improving recall should be the primary objective.

The information sources described in Chapter 4 proved informative for both tasks, with features being selected from all categories. Performance improved after feature selection, and stratified feature group selection was typically the optimal strategy. Hyperparameter optimization improved the results for relevance, not for severity.

The experiments with cascades revealed that in our setup, they do not provide a performance benefit over direct classification in terms of F_1-score. It does, however, allow the use of TiMBL, which performs better than the direct and cascaded SVM systems in terms of recall and F_2.

In the next chapter, we venture to improve detection performance by addressing linguistic noise, a typical phenomenon in user-generated content, with automatic text normalization.

CHAPTER 8

Normalization

In this thesis, we investigate the feasibility of automatically detecting suicidal content in social media. A distinctive characteristic of such user-generated content is that it tends to deviate from the linguistic norm. Typical problems include the use and productivity of abbreviations, deliberate misspellings, phonetic text, colloquial and ungrammatical language use, lack of punctuation and inconsistent capitalization.

These abnormalities may hinder automatic text processing. Many state-of-the-art text processing tools are available for Dutch and other languages, but they have all been developed with standard text in mind. As a result, a significant drop in performance can be observed when they are applied to user-generated content. This is for example the case when applying parsing (Foster et al. 2011) or named entity recognition (Liu, Shaodian Zhang and Zhou 2011, Ritter et al. 2011) to Twitter data.

In Chapter 4, we discussed how noise in our data affects the accuracy of tokenization and lemmatization, two preprocessing steps that are necessary for feature extraction. To improve preprocessing accuracy, we applied rules to remove or correct a number of uniform noisy occurrences, such as emoticons, URLs, tags, or missing spaces. This had some positive effect, but it only cor-

rected the 'low-hanging fruit', i.e. the errors that were consistent enough to be handled with fixed rules. For example, any misspelled words would remain unchanged, and their lemmas would likely be incorrect. A majority of the extracted feature groups rely on word and lemma information, either directly (the word and lemma ngram bag-of-words), or through matching (the lexicon, TERM and LSA features). Noise affects both categories. Bag-of-words features will become sparser, as there may be multiple variants (i.e. misspellings) of the same word or lemma. Noisy forms will also fail to match the entries in a lexicon, term list or topic model. The experimental results in the previous chapter demonstrated the importance of many of these feature groups. We hypothesize that improving lexical recall may help in the important objective of improving overall classification recall, as underlined in Chapter 7. In this chapter, we therefore investigate how noise in user-generated content can be reduced.

The task of transforming noisy input into its standard equivalent is known as text normalization. Currently, no systems are publicly available for the normalization of Dutch text. We therefore collected and annotated three different types of user-generated content (text messages, message board posts and tweets), a corpus presented in De Clercq et al. (2014), and developed a normalization system that uses machine translation, described in De Clercq et al. (2013). We test a traditional token-based translation, and combine it with a novel character-based translation approach.

In Section 8.1, we discuss the current state of the art in normalization research. Section 8.2 describes the corpora and annotations that were used in our study, and the architecture of the proposed system. The experimental results are presented and analysed in Section 8.3. We summarize the work on normalization in Section 8.3.5, and formulate perspectives for future work. Finally, in Section 8.4, we apply the obtained normalization system to the data for the suicide tasks, and investigate how this affects feature extraction and classification performance.

8.1 Related research

Traditionally, the task of text normalization is a crucial first step for every text-to-speech system, in which specific numbers and digit sequences, acronyms, etc. need to be rewritten in order to have them pronounced correctly. A thorough overview of the main characteristics and bottlenecks can be found in Sproat et al. (2001).

More recently, however, the surge of social media has introduced a range of new problems stemming from its noisy content. This reality, combined with the

need to process data from social media, has revived the interest in normalization techniques. In this regard, we can define three dominant approaches to convert noisy into standard text. These are referred to as the spell-checking, machine translation and speech recognition metaphors (Kobus et al. 2008).

The most intuitive way of normalizing text would be to approach the problem as a spell-checking one, where noisy text has to be transformed to standard text using noisy channel models. Choudhury et al. (2007), for example, proposed a supervised noisy channel model using Hidden Markov Models to calculate the probability of less frequent words. Extensions to this approach were made by studying word processes (Cook and Stevenson 2009), adapting weighted finite-state machines and rewrite rules (Beaufort et al. 2010) or by adding other elements such as orthographic, phonetic and contextual factors (Xue et al. 2011).

Another approach is using statistical machine translation (SMT) techniques for text normalization. These approaches cast the normalization problem as one where a noisy text should be translated into a clean text, as if it were a different language. For standard machine translation between languages, SMT is currently the most successful approach. Previous work that applies SMT techniques to normalization has mostly focused on phrase-based MT at the word level. Aw et al. (2006) were the first to compare its performance on English SMS to dictionary substitution using frequencies, and found that SMT performed better. Also working on English text, Raghunathan and Krawczyk (2009) confirmed that using an SMT system outperforms dictionary lookup, most notably when used on an out-of-domain test set.

Kobus et al. (2008) followed the same approach but combined the machine translation features with a speech recognition system using HMMs on a French corpus. Speech recognition techniques were used to convert the noisy input into phonemes, and then converting them back into graphemes using a language model. This approach should allow a system to normalize misspelled words that are written phonetically. They concluded that the two systems perform better on different aspects of the task, and that combining the modules works best.

An entirely different way of approaching normalization is the work by Liu et al. (2011, 2012). They propose an unsupervised cognition-driven text normalization system. By observing and simulating human techniques for the normalization task, they avoid dependence on annotated data. They construct a broad-coverage system to enable better word coverage, using three key components: enhanced letter transformation, visual priming and string/phonetic similarity.

Intuitively, the normalization task has a lot in common with transliteration

tasks, e.g. between scripts or between historical versions of a language. For transliteration, character-based SMT systems have proven adequate (Vilar et al. 2007). Pennell and Liu (2011) were the first to study character-based normalization. However, they only applied their approach for abbreviation expansion.

We propose a cascaded SMT model, in which token-based and character-based normalization is combined.

8.2 Methodology

In order to build a supervised SMT system for normalization and evaluate its performance, it is essential to construct a gold standard dataset that can serve as training and test material. This involves compiling a corpus (8.2.1), and developing and applying guidelines for normalization (8.2.2). The system architecture is described in Section 8.2.3.

8.2.1 Corpus compilation

To ensure that our corpus is representative of the domain of user-generated content (UGC), we decided to include three different social media genres: text messages (SMS), message board posts from a social networking site (SNS), and tweets (TWE). For the SMS genre, we sampled 1 000 messages from the Flemish part of the SoNaR corpus (Treurniet et al. 2012). The sampling procedure aimed at a balanced spread of two author characteristics: their age and region. For the SNS genre, which allows longer messages, 1 505 message board posts were randomly selected from a Netlog corpus, distinct from the ones used in the suicide experiments. Given the intense focus of normalization research on Twitter data, we also included 246 randomly selected tweets. It is to be noted, however, that in general, Twitter content in Belgium differs from that in English-speaking countries or the Netherlands, since it has mainly been adopted amongst professionals. As a result, it does not exhibit the amounts of noise described in related research on English tweets. Example messages are presented in Table 8.1, in the top row for each genre.

The examples illustrate the main characteristics of Dutch UGC. The problems that are encountered are mostly similar to those that have been described for other languages, such as English (Baron 2003) and French (Beaufort et al. 2010).

Some of the more well-known problems include the omission of words or characters, e.g. the omission of the final *n* in *gesproke* (EN: *spoke* versus *spoken*).

SMS	original	Oguz ! Edde me Jana gesproke ? En ze flipt lyk omdak ghsmoord heb .. !
	normalized	Oh gods ! Heb je met Jana gesproken ? En ze flipt gelijk omdat ik gesmoord heb ... !
	English	Oh god ! Did you speak to Jana ? And she's flipping because I smoked ... !
SNS	original	schaaaat , Je komt wel boven die Blo , je et em nii nodig wie jou laat gaan is gwn DOM :p Iloveyouuuu hvj
	normalized	schat , Je komt wel boven die Blo , je hebt hem niet nodig wie jou laat gaan is gewoon dom :p I love you hou van je
	English	honey, You'll get over that Blo, you don't need him whoever lets you go is just stupid :p I love you I love you
TWE	original	@minnebelle top ! Tis voor m'n daddy !
	normalized	@minnebelle top ! Het is voor m'n daddy !
	English	@minnebelle great ! It is for my daddy !

Table 8.1: Examples from the three social media genres under study, representing the original utterance, its normalized version and an English translation.

Abbreviations and acronyms also occur frequently (e.g.*gwn, hvj*, EN: *LOL*), and they are highly productive. Moreover, many utterances deviate from the standard spelling because they are written as they are pronounced (e.g. *lyk* instead of *gelijk*, EN: *luv* instead of *love*), or because of colloquial language (e.g. *et em* instead of *hebt hem*, EN: *you iz* instead of *you are*). Additionally, emotion can be expressed or emphasized by using flooding (repetition of the same character or character sequence, e.g. *baaaaaaby*), emoticons (*:p*) and capitalization (*STUPID*).

More specific to the Dutch language[1] is the reduction and concatenation of tokens, typically pronouns (e.g. *Edde* instead of *Heb je*, *khou* instead *ik hou*, *Tis* instead of *Het is*). Moreover, the widespread influence and adoption of English, and the fact that Belgium is a trilingual country, may lead to various languages being used within a single utterance (*Oguz, daddy, we are forever*). For an in-depth study of the linguistic variation found in online Dutch, we refer to the study on teenagers' chat language by Vandekerckhove and Nobels (2010).

[1]This phenomenon also appears in English (e.g. *gimme* or *gonna*), but it is restricted to a small number of fixed cases.

8.2.2 Annotation

All text material was annotated by two annotators, independently of each other, using newly developed normalization guidelines. These guidelines, tailored for Dutch, have been drawn up in close collaboration with the developers of the Chatty Corpus (Kestemont et al. 2012).

The guidelines describe an annotation procedure that can roughly be divided into two parts. The first part consists of the actual normalization, and comprises three steps: correcting obvious tokenization problems, providing a fully normalized version, and stating the different normalization operations necessary to go from an original to a normalized token. We allow four different operations:

- Insertion of missing characters (INS), e.g. *spoke* → *spoken*, *sis* → *sister*
- Deletion of superfluous characters (DEL), e.g. *baaaaabyyyy* → *baby*
- Substitution of one character (sequence) for another (SUB), which is equivalent to one or more deletions combined with one or more insertions at the same location, e.g. *iz* → *is*, *stoopid* → *stupid*
- Transposition, where the order of two characters is switched (TRA), e.g. *liek* → *like*

The second part consists of flagging additional information that might be useful for automatic processing purposes. Annotators were asked to indicate the ends of thoughts in an utterance, to compensate for missing punctuation. They also flagged regional or foreign words, named entities, grammatical errors, and words that are stressed, part of a compound, used as interjections, or that required consecutive normalization operations.

To check the reliability of our annotation guidelines, the two annotators each normalized the 1,000 text messages. We estimated the inter-annotator reliability by computing word error rate (described below) between the two fully normalized versions. The WER was 0.048, which indicates near-perfect overlap.

Genre	**#**	**Before**	**After**	%	**INS**	**DEL**	**SUB**	**TRA**
SMS	1 000	16 630	17 194	3.39	3 622	338	547	57
SNS	1 505	31 513	32 221	2.25	4 165	1 500	1 692	57
TWE	246	3 276	3 357	2.47	923	67	127	4

Table 8.2: Normalization statistics of the three UGC genres. On the left: message and token counts; on the right: operation counts.

To give an intuition of the normalization effort required, we present some statistics for each genre in Table 8.2. The left-hand side shows the number of tokens before and after normalization and the relative change in percent. On the right-hand side, the normalization effort is expressed as the number of each of the four operations. Examples of normalized text can be found in Table 8.1, with normalizations in the middle row for each genre, and English translations in the bottom row.

For the experiments in this study, we work with the first part of the gold standard normalizations (ignoring flagging information such as ends of thought). We focus on the SMS data, because it is the noisiest genre in our corpus, with a token increase of 3.39%.

8.2.3 System architecture

As was already mentioned in the related research section, statistical machine translation can be used for text normalization at various levels of granularity. At the token level, an SMT system can outperform a simple dictionary lookup for the 'translation' of high-frequency words and abbreviations, because it can translate them in context. SMT models work by extracting pairs of source segments and their translations from an aligned parallel corpus (in our case, the aligned original and normalized tokens), and collecting them in a phrase table. During translation, the input is matched to the source segments in the phrase table, all possible translations are retrieved, and a language model is used to find the most probable combination. Language models are statistical ngram models, derived from a reference corpus, which allow to estimate the probability of a word or word sequence, given the preceding word(s). We can also apply the SMT approach to the character level, which would allow a system to learn typical character mappings. This can make a system much more robust to unseen variation (Pennell and Liu 2011). The omission of the character *n* at the end of verb forms (as in *gesproke*) provides a good example of why working on the character level can be advantageous. For a token-based translation model to be able to successfully normalize a different verb (e.g. *spreke*), it would need to have access to an example of that specific form in the training corpus. A character-based translation model, on the other hand, would be able to infer the character mapping from *e* to *en* in specific contexts, and apply it to unseen verb forms.

Prior to any sort of learning, we adapted our tokenizer to be able to handle emoticons, hyperlinks, hashtags and at-replies. To that end, rewrite rules were devised, similar to Beaufort et al. (2010). We also decided to tackle the flooding of characters before applying machine translation, in order to reduce potential

confusion. Characters and character sequences were allowed to occur a maximum of two times consecutively. Any higher number of repetitions was reduced to two. The validity of this approach was checked by applying this rewrite rule on the CELEX database (Baayen et al. 1995), which contains 381 292 valid Dutch words, including inflections. A mere two (highly infrequent) entries were changed by the rule, which confirms that it virtually does not overnormalize.

After this preliminary preprocessing, the noisy text is processed by the SMT models, either individually, or in a cascade. In cascades, the standard phrase-based SMT approach at the token level is first used to ensure the translation of the more frequently used abbreviations (such as *fb* for *facebook* or *coz* for *because*). Afterwards, the translated text is split into characters and a translation at the character level takes place. We test character unigram and bigram translation models. Bigrams supposedly have the advantage that one character of context across phrase boundaries can improve the selection of translation alternatives from the phrase table (Tiedemann 2012). This means that more precise translations will be suggested.

We first focus on the performance that can be achieved within the SMS genre, and then test the approach on the other genres to see whether it is robust. For evaluation, both the Word Error Rate (WER) and BLEU scores were calculated. WER is an evaluation metric that is based on edit distance at the word level. It is very well suited for the evaluation of NLP tasks where the input and output strings are closely related. This is why the metric is typically used for the evaluation of optical character recognition (Kolak et al. 2003), grapheme-to-phoneme conversion (Demberg et al. 2007), diacritization (Schlippe et al. 2008) and vocalization of Arabic (Kübler and Mohamed 2008). The BLEU metric, which has been specifically designed for measuring machine translation quality, measures the ngram overlap between the suggested translation and a set of gold standard translations. We believe that BLEU is less appropriate for the evaluation of normalized output, but we include it for comparison's sake, as it has been reported for other systems (Aw et al. 2006, Kobus et al. 2008).

8.3 Experiments and discussion

8.3.1 Experimental setup

For all experiments, we used Moses, a state-of-the-art open-source SMT system (Koehn et al. 2007). As a reference corpus for the language model, we used the Spoken Dutch Corpus (Corpus Gesproken Nederlands, CGN (Oostdijk 2000)) since spoken language would better reflect the language encountered in UGC.

All language models were built using the SRILM toolkit (Stolcke 2002) with Witten-Bell discounting, which has been proven to work well on small data sets (Tiedemann 2012).

We trained a number of translation models. The token-level translation model was built using the standard Moses settings and a 5-gram language model. For the character-level model, the same Moses settings were used, but we experimented with different sizes of n for the language models (5, 7, 10 and 15). We found that the 10-gram language model gave the best results.

For the first set of experiments (8.3.2), training was performed on the SMS data, which was divided into three samples: 625 messages for training, 125 for development and 125 for testing. In order to estimate the system's robustness to unseen genres, the SMS-tuned system was then tested on the other two genres, 125 SNS posts and 125 tweets (8.3.3).

8.3.2 Results on SMS

Setup	**SMS**		**SNS**		**TWE**	
	WER	BLEU	WER	BLEU	WER	BLEU
A. Original	21.70	65.54	20.41	66.03	13.26	76.10
B. Baseline	21.47	65.64	20.36	65.93	13.26	76.10
C. Token-level only	20.41	76.04	25.03	73.26	19.03	78.32
D. Unigram only	14.93	66.45	15.41	64.02	13.52	66.29
E. Bigram only	15.90	64.26	15.17	63.94	14.08	65.50
F. Cascaded unigram	**13.11**	69.48	**14.59**	65.17	**10.35**	72.25
G. Cascaded bigram	14.65	66.55	**14.59**	64.79	10.36	72.25

Table 8.3: WER and BLEU performance of the seven setups, trained on the SMS genre and tested on all genres.

All experimental results on the SMS data, using various setups, can be found in Table 8.3. They are expressed both in terms of WER (where lower is better) and BLEU (where higher is better). Figure 8.1 presents a visual overview of the performance in terms of WER. Since there is no prior work on Dutch normalization, there is no basis for comparison to other systems. We start by reporting the difference between the original source and target text (A) as well as a baseline where only the rewrite rules have been applied (B). We notice that a moderate improvement in WER, from 21.70 to 21.47%, already occurs by eliminating flooding.

Next, the various SMT models were tested. These results all clearly outperform

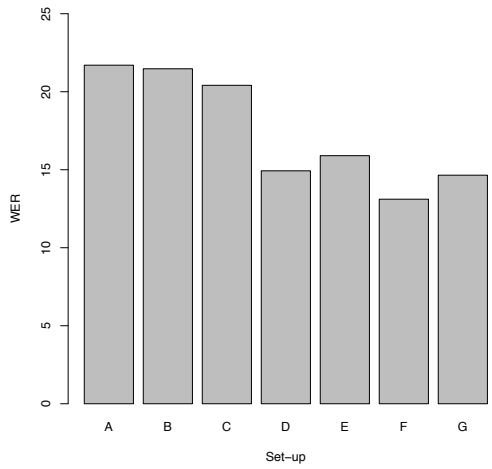

Figure 8.1: Visualization of the WER reduction on the SMS data set using seven different setups.

the baseline. The token-based model (C) accounts for a moderate improvement, but the character-based models, both with unigrams (D) and bigrams (E), perform much better. When both approaches are combined in a cascade, results further improve, both with the unigram (F) and bigram (G) cascades. The best result is obtained with the cascaded unigram model (F). This model has a WER of 13.11, which is a 63% drop in word error rate over the baseline and 56% over the non-cascaded word level SMT.

When the same analysis is performed using the BLEU evaluation metric (Figure 8.2), we observe a different tendency. The token-based model (C) clearly achieves the best score, whereas the cascaded unigram model (F) only achieves the second best result. This could be explained by the inherent differences between the metrics. WER is based on edit distance whereas BLEU measures n-gram overlap. This means that the output of the unigram cascaded model can be closer - but not perfect - to the gold standard than the output from the token model. Considering Example 37 below, we see that the token model is not able to find the correct version, whereas the output from the cascaded unigram model is already a step in the right direction. If we would then feed this closer version back into our token model, it should be able to resolve it correctly. This insight could be used to further improve our system by extending the cascaded unigram module with another run of the token-based system.

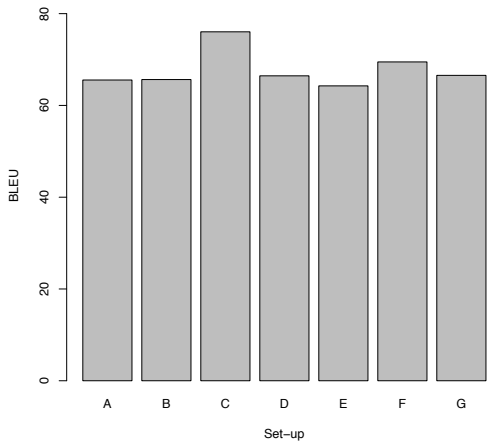

Figure 8.2: Visualization of BLEU on the SMS data set using seven different setups.

(37) *original:* laatk → *target:* laat ik
output C: laatk → *output F:* laat k → *output C:* laat ik

8.3.3 Results on all genres

The results of applying the normalization models tuned for SMS to the other two genres are available in Table 8.3. These results provide insight into the robustness of each approach.

Applying the baseline system with rewrite rules gives the same minor positive effect for SNS as for SMS, compared to the original source and target text. For tweets, on the other hand, no improvement is noted. Upon closer inspection of the Twitter data, not a single instance of flooding was found, which explains this status quo.

When comparing the other models, the same evolution in word error rate can be observed. For each genre, the best WER reduction over the baseline is reached with the cascaded unigram model, namely 63% for SMS, 39% for SNS and 28% for TWE. For the SNS data, the cascaded unigram and bigram translation models attain equal performance. Again, the BLEU metric favors the token-based translation model, for each genre. This trend is most notable for the

Twitter data, where abbreviations are frequent.

8.3.4 Error analysis

A qualitative error analysis was performed on the output of the best approach in terms of WER, i.e. the cascaded unigram approach. Inspection of the SMS test output revealed that the system was able to locate and resolve 172 of the 320 tokens requiring normalization. The system also generated 51 false positives, which leads to a precision of 77.13%, a recall of 55.66% and an F_1 score of 64.66%.

Operation	**Gold standard**	**# missed**	**% missed**
INS	549	270	49%
DEL	28	20	71%
SUB	55	30	54%
TRA	11	6	54%

Table 8.4: Absolute and relative number of missed operations at the character level.

We categorized the false negatives in two ways. First, we investigated which types of operations (cf. Section 8.2.2) proved to be the most difficult to resolve. Since a token may need multiple or different operations[2], this was calculated at the character level. Table 8.4 presents the number of operations missed by our system, both in absolute and relative numbers.

At first sight, the deletions seem especially hard to resolve, with 71% of the cases missed, followed by substitutions and transpositions. When the absolute numbers are taken into account, however, these classes are proportionally much less frequent than the total number of needed insertions (549). The system appears to be able to resolve around half of these (51%). On closer inspection, we find that the system is especially good at normalizing shorter words requiring only one or two insertions, such as *eb* for *heb*, *nie* for *niet*, and not in building longer words such as *gr* for *groetjes*. If we consider the number of normalizations involving insertions at the word level, rather than at the character level, we indeed find a higher success rate, with 60% of cases being successfully resolved. Another observation at the word level is that corrections requiring different types of operations are hard: only 44% are successfully replaced.

These observations make intuitive sense. Insertions are the most common op-

[2]For example *sis* → *sister* requires three insertions, and *luv* → *love* requires a substitution and an insertion.

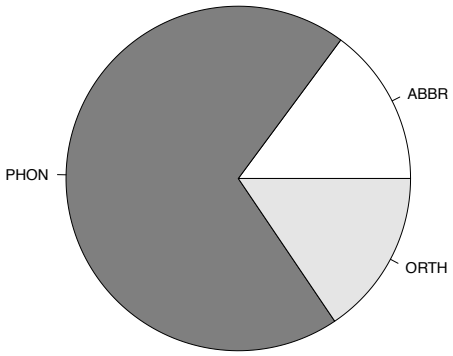

Figure 8.3: Proportion of missing normalization per error category.

eration in the training data as well, so translation models have access to more examples that require insertion. The drop in performance on longer transformation sequences is likely also caused by data sparsity. The *gr* → *groetjes* correction can only be handled by the token-based model, which depends on exact matches in the training data to replace abbreviations. Likewise, the character-based model would need more examples of transpositions, substitutions and deletions to be able to correct them, especially if they are combined.

Class	**Output**	**Correct**	**Translation**
ABBR	aug	augustus	August
PHON	hebk	heb ik	have I
ORTH	uan	van	of

Table 8.5: Examples of missing normalizations of each error category.

The second error classification takes a more linguistically motivated approach. Inspired by the work of Androutsopoulos (2007), we define three error types: abbreviation (ABBR), phonetic (PHON) and orthographic (ORTH) issues. Table 8.5 lists an example of each of these categories, that was missed by our system.

In Figure 8.3, the proportion of errors per category is visualized. We find that resolving phonetic problems is the main issue for our system, with 103 missed instances. Closer inspection of the phonetic errors revealed the following phenomena in the instances that were missed: fusions (concatenations of words, 25%), omissions (missing characters, 43%), homophones (characters referring

to the same sound, 26%) and onomatopoeias ('sounds like', 6%). Especially the omission of characters seems problematic, which is consistent with the high number of missed insertions (i.e. 270 characters).

These errors indicate that our system might benefit from including other modules, apart from machine translation. The orthographic issues might probably be resolved using a spell checker, whereas the phonetic ones, especially the homophones, might benefit from grapheme-to-phoneme conversion.

The system also generated 51 false positives on the test data, which can be considered as hypercorrections. Fifteen of these are named entities or foreign words, which should not be normalized. It may be worthwhile to extend the preprocessing pipeline, so that these words can be detected and protected from normalization further down the line.

8.3.5 Summary

We have developed a statistical machine translation approach to normalize Dutch user-generated content (UGC), and tested it on three genres. The experiments on SMS messages, the genre requiring the most normalization, revealed that a cascaded model, in which a token-based module is followed by a translation at the character level, yields the best results. Tests on the other genres confirmed this, which indicates that the approach is robust across genres.

We see a number of possibilities to improve normalization performance. A logical first step would be to increase the amount of training material. We believe that performance might also benefit from an approach in which a number of diverse modules are combined, each of which tuned to specific normalization problems. Considering the error analysis, we feel that modules inspired on the three metaphors (machine translation, spell checking and speech recognition) might produce a strong combination, capable of tackling a wider range of problems than our current setup. Furthermore, it might be worthwhile to investigate the effectiveness of passing normalized content through the system a second time, to allow modules to further enhance the output of other modules.

8.4 Normalization and suicide detection

In the previous sections, we described the development and evaluation of a normalization system for Dutch user-generated content. Now, we apply this system to the suicide data, extract features based on the normalized text, and repeat a number of optimization experiments using these new features. The findings should be interesting from two perspectives.

First, it provides an extrinsic evaluation of the normalization system. Typically, normalization is not an end in itself. It can be, e.g. for improving the readability of a text, but more often than not, it improves the input for another task (e.g. text-to-speech synthesis), or as a component in a preprocessing chain, such as in our experiments. In the experimental section above, we have intrinsically evaluated normalization effectiveness by comparing the system's output to a gold standard solution, and calculating the WER and BLEU measures. By comparing systems for the suicide detection tasks with and without normalization, we can extrinsically evaluate whether normalization contributes to better performance.

From the perspective of the relevance and severity tasks, these experiments can shed light on whether the added complexity (and computational overhead) of doing normalization is justifiable, and to which extent the original system was robust to noise.

8.4.1 Normalization of the data

We opted to use the best-performing normalization approach in terms of word error rate, i.e. the cascade of a token-based and a character unigram-based SMT model. The models were retrained on the full gold standard dataset containing all three genres, and were applied to normalize the development set and held-out testset for the suicide tasks.

Normalization took an average of one minute per post, which is why we did not normalize the scaling datasets. The current computational cost of normalization would be fairly prohibitive to applying normalization in a real-world application. It should be noted, however, that most messages were normalized in a fraction of the time. The Moses character models took a disproportionately large amount of time to normalize very long sentences, which drove up the average processing time. This is not surprising, since Moses aims to find the most probable sequence of phrase table suggestions for an entire sentence. This problem grows in complexity with sentence length, and with characters-based translation, the number of segments is significantly higher than with tokens. A

workaround for this problem would be to split sentences into shorter sections. This is fine for most characters, because models will not take into account context that is more than 10 characters away, but it removes context around the characters near a split.

An example of a full Netlog post before and after normalization is presented in Table 8.6. The following problems are italicized:

- Errors introduced by normalization, e.g. *Boo*, a named entity that should have remained unchanged, *wha*, which is converted to the English word *what* rather than the Dutch homophone *wat*, or the apostrophes in the final sentence that are changed into the contracted pronouns *'t* and *'m*.
- Missing normalizations, such as the possessive pronoun *U* which should have been written as *uw* (although this can be considered a grammatical error), *bzg* instead of the correct *bezig* (EN: *busy*) or *benen*, which is a concatenation of *ben en* (EN: am and), which is missed because in Dutch, *benen* (EN: *legs*) is a valid word.
- Unintelligible source segments that cannot be normalized, e.g. *ua*, *n.n*, which seems to be used as an emoticon, or *hewhiiii*, which was reduced to *hewhii* by the flooding rule and then normalized to *hewhi*.

These errors affect a number of content words (*bezig*, *ben en*), but we find that the majority of noisy content words are correctly normalized:

- *gwn* → *gewoon* (EN: *just*)
- *gha* → *ga* (EN: *go*)
- *loslatten* → *loslaten* (EN: *let go*)
- *goe* → *goed* (EN: *good*)
- *wissele* → *wisselen* (EN: *change*)
- *kzen* → *ik zijn* (EN: *I be*)

In the context of text categorization, where content words typically are much more informative than function words, we can assert that our normalization system successfully reduces noise in the words that matter, and introduces errors that are likely to be of minor importance.

Original	**Normalized**
ua,	*ua* ,
ik zal maar is Beginnen	ik zal maar is beginnen
Boo jij gaat echt voor altijd in mijn hartje zitten	*bo* jij gaat echt voor altijd in mijn hartje zitten .
Jij gaat er nooit meer uit	jij gaat er nooit meer uit
jij bent **gwn** een droom voor mij	jij bent **gewoon** een droom voor mij
die ik nooit **gha loslatten** of **gha** laten vliegen	die ik nooit **ga loslaten** of **ga** laten vliegen
jij bent zo LEKKER ;	jij bent zo lekker ;
oals ik u zie **gha** ik zo **goe** vastnemen en Fluister	**als** ik u zie **ga** ik zo **goed** vastnemen en fluister
“ik hou mega veel van je”	ik hou mega veel van je
U sexy Blauw broekje *n.n*	*u* sexy blauw broekje *een n* .
als we elkaar zien,	als we elkaar zien ,
wissele we van broek *hewhiiii*,	**wisselen** we van broek *hewhi* ,
jua kzen nu zo *bzg* ***wha*** ik allemaal **gha** doen *medu*	**ja ik zijn** nu zo *bzg* ***what*** ik allemaal **ga** doen *mede*
als ik ooit depri *benen* zelfmoord pleeg	als ik ooit depri *benen* zelfmoord pleeg
dan gebeurt er dit;	dan gebeurt er dit ;
als ik mij neerschiet	als ik mij neerschiet
dan moet je weten	dan moet je weten
dha jij de laatste bent **dhie** in mijn hoofd zat	*had* jij de laatste bent **die** in mijn hoofd zat
want op de kogel staat	want op de kogel staat
‘Boo, ik hou van je*’*	*‘t boo* , ik hou van je *‘m*

Table 8.6: Example of a Netlog post before and after normalization. Tokens with correct normalization edits are boldfaced, missing or incorrect normalizations and unsolvable cases are italicized.

8.4.2 Feature extraction

New feature vectors had to be constructed, based on the normalized texts. As discussed in Chapter 4, features are calculated on one or more layers: the *original* layer, the *clean* layer (cleaned with rewrite rules), the *last* layer (containing only the last 10 tokens) and the *normalized* layer. All features[3] that had previously been based on the clean layer were now recalculated on the normalized layer. This affects the W, LEM, LCH ngram features, the PAT and DUO lexicon features, and all TERM and LSA features.

	original	**normalized**	**relative change**
W1	51 670	34 867	-33%
W2	300 277	219 102	-27%
W3	475 167	391 030	-18%
LEM1	50 739	33 505	-34%
LEM2	289 438	205 589	-29%
LEM3	481 639	391 598	-19%
LCH1	1 799	1 743	-3%
LCH2	18 136	16 070	-11%
LCH3	92 049	72 932	-21%
All features	1 934 560	1 539 708	-20%
After relevance feature filtering	21 791	22 144	2%
After severity feature filtering	9 351	10 068	8%

Table 8.7: Number of features, based on original or normalized text.

In Table 8.7, we list the feature counts for the original vectors, and the ones using the normalized text layer. Feature counts could only change for the ngram features, which are listed individually. We find that normalization is very successful in reducing linguistic variations. For the token-based feature groups, dimensionality drops substantially, with between a fifth and a third of the features removed. The effect is best observed on the unigram features: of the original 50 000 word forms, around 17 000 variants are removed. Not surprisingly, the effect is less pronounced for the character ngram features, since the amount of potential character combinations is more limited.

Overall, the feature vector length is reduced by 20%. For systems where no feature filtering is applied, this would be a significant advantage. As in the original experiments, we applied the information gain filtering metric for each task, with the same threshold of 0.001. Interestingly, the number of informative

[3]With the exception of the NE features. Named entities counts had been extracted on the basis of the clean layer, and the same counts were used in the normalization experiments, since named entities are often mangled by normalization.

features increases because of normalization, by 2% for the relevance task and 8% for severity. From this, we can conclude that normalization has improved lexical recall, as intended.

8.4.3 Classification tasks

To compare the performance of models that use the original feature vectors to those using normalized data, we repeat two experiments and analyse the results. For each task, we reapply the settings of the best-performing model after F_1 optimization in an optimization run with normalized features. For relevance, joint optimization with stratified feature group selection yielded the highest F_1 score (92.55%). For severity, a regular feature group selection strategy without hyperparameter optimization scored best at 69.04%.

Relevance

dataset		F_1	F_2	precision	recall
development	original	92.55	92.59	**92.50**	92.61
	normalized	**93.33**	**94.15**	91.99	**94.70**
	difference	0.77	1.56	-0.51	2.09
	error reduction	10.35%	21.01%	-6.80%	28.28%
held-out	original	**93.51**	**91.37**	**97.30**	90.00
	normalized	91.36	90.24	90.24	**92.50**

Table 8.8: Comparison of results on the relevance task, with and without normalization.

The results of the optimization run for the relevance task using normalized vectors are presented in Table 8.8. The model with normalization performs markedly better on the development set. Precision is slightly lower than that of the original model, but the improvement in recall constitutes an error reduction of almost 30 percent. At 94.70%, it is the highest recall obtained by any model. Consequently, the model also performs best in F_1 and especially F_2. As was already suggested in the qualitative error analysis of the original relevance model (Section 7.5.1), an improvement in recall is more desirable than better precision.

On the held-out testset, we observe the same tendency towards recall. None of the original models could retrieve 37 of the 40 relevant messages. However, compared to the development set, the trade-off in precision is higher. Still, these results are promising, and given its high recall, the normalized model would likely be a superior candidate as a first step in a cascaded severity model.

Severity

dataset		F_1	F_2	precision	recall
development	original	69.04	63.93	79.67	60.92
	normalized	**71.91**	**66.58**	**82.97**	**63.45**
	difference	2.87	2.65	3.30	2.53
	error reduction	9.27%	7.35%	16.23%	6.47%
held-out	original	**66.67**	**59.14**	**84.62**	**55.00**
	normalized	43.48	47.17	38.46	50.00

Table 8.9: Comparison of results on the severity task, with and without normalization.

The effect of normalization on the results for the severity task is displayed in Table 8.9. After optimization on the development data, the normalized model scores better than the best original model in every respect. It achieves the overall highest F_1 score (71.91%). Some other models perform better in terms of recall and precision, but are typically unbalanced towards one or the other.

On the held-out testset, the normalized model is among the better performers in terms of recall (1 in 2 severe messages detected), but it fails to match the original model's performance. Precision is much lower. It seems that systems that were optimized to be less conservative in their predictions score well on the development set, but overgenerate on the held-out data, without achieving better recall. We can observe the same phenomenon with the original F_2-optimized severity models (page 124). This effect may be specific to the positive instances in the held-out sample, the small number of which exaggerates the weight of idiosyncrasies.

8.4.4 Summary

In this section, we investigated normalization in the context of the suicide detection tasks. A qualitative inspection of the output from the normalization pipeline showed that it adequately reduced noise in the original texts, mainly in content words. It also created a small number of undesired artefacts, most of which would not hinder text classification.

The application of normalization for the suicide detection tasks has demonstrated its benefits. We observed a significant dimensionality reduction in the full feature vectors, and increased feature informativeness. In the classification experiments, normalized models clearly outperformed the original ones, if only on the development data. Normalization improved recall on the relevance task

in particular.

We can conclude that as an extrinsic evaluation, these experiments have evidenced the utility of normalization. There is ample potential for further research, both on normalization and on how it is best exploited to improve the performance of text classification on noisy user-generated content.

CHAPTER 9

Conclusion

This study set out to investigate the automatic detection of suicide-related posts on social media. Online platforms have become an important means of communication, and they may be used as an outlet to express suicidal thoughts. Peers and service administrators may notice or respond to suicidal messages too late, if at all, and the information overload associated with social media impedes suicide prevention stakeholders from successfully monitoring for them.

Research on the automatic detection of suicidal content has so far been limited. It focused on keyword-based search approaches, which have a number of disadvantages. Compiling a list of reliable search terms is typically a manual endeavour, and terms must be sufficiently specific to avoid irrelevant results. Consequently, they can only find references to suicide that are lexicalized explicitly and with known expressions. We apply natural language processing and machine learning techniques to approach the problem from a text classification perspective. The main contribution of this thesis is that it presents the first study on classification-based suicidality detection, in Dutch user-generated content.

9.1 Resources

The first research objective, described in Chapter 3, was to create an essential resource for training and validating supervised classification models: an annotated corpus containing suicide-related material. We found that acquiring such material for annotation is not trivial: suicidal messages are very scarce in randomly collected user-generated content, and there are currently no automatic means of detecting them. The collection method was therefore based on keyword searches and manual selection. Two sets of blog and forum posts were obtained from the social networking site Netlog: a corpus of 1 040 messages that were deemed suicide-related, which were all manually annotated, and a 300 000-post reference corpus.

For the annotation of suicidality, a scheme was developed that is grounded in suicide prevention practice. It provides for the annotation of a post's relevance to suicide, its genre, and the subject, severity and explicitness of a suicide threat, if any. From this, the relevance and severity classification tasks were derived. The scheme was tested in an inter-annotator agreement study, which demonstrated that it allows reliable annotation. Annotation of severity is found to be inherently ambiguous: there are no infallible protocols for diagnosing suicide ideation, and the limitations of the medium further complicate diagnosis.

9.2 System architecture

Chapters 4, 5, 6 and 8 discussed the proposed methodology for building, validating and optimizing models for suicidality detection.

We elaborated on feature vector construction in Chapter 4. Bag-of-words features are the most well-known information source for text classification systems. We calculated token, lemma and character ngrams up to length 4 as unweighted, binary features. With the LSA features, we aim to group semantically related words together, in order to abstract away from the word level and reduce vector sparsity. Term matching was used to flag highly relevant terms that were extracted from two suicide-related background corpora. We calculated the polarity orientation of posts and their final words using a number of lexicons developed for sentiment mining research. Lastly, miscellaneous features describing post length, capitalization and the occurrence of named entities were added.

These features are combined with two supervised learning algorithms: LIBSVM, a support vector machine implementation, and TiMBL, a memory-based learner. Chapter 5 described the learning hypotheses and hyperparameters of these al-

gorithms, and laid out the methodology for model validation: we measure performance with the $F_{\beta=1}$ and $F_{\beta=2}$ metrics, using tenfold cross-validation. Additionally, it introduced cascades as an alternative to direct classification for the severity task.

Given these information sources and learners, we sought to optimally exploit their discriminative power by optimizing the models. Optimization is aimed at finding a good combination of relevant features and robust algorithm settings. We performed feature filtering based on information gain, reducing the number of bag-of-words features by 99% and other features by 50%. Three strategies were defined for wrapped feature group selection: *regular* (full) feature groups, *nbest* groups limited to only include the 500 features with the highest information gain, and *stratified* feature group selection, which allows the selection of subsets (strata) of a feature group. To solve the search problems of hyperparameter optimization, feature selection and joint optimization, we used Gallop, a genetic algorithms implementation developed to run computationally expensive experiments on distributed supercomputer infrastructure.

To counteract the problem of noise in user-generated content, we developed a system for automatic normalization (Chapter 8). Experiments on three genres (SMS, tweets and forum posts) showed that a cascaded statistical machine translation approach yields the best results: after correcting flooding (superfluous repetitions of character sequences) with a rewrite rule, a token-based SMT module replaces common abbreviations and mistakes, followed by a translation at the character level to correct small orthographic and phonetic variations. The latter step makes the system more robust, because it can generalize and apply character-level transformations to unseen words. Overall, we notice a significant drop in word error rate after applying normalization. We hypothesized that normalization can reduce vector sparsity and improve performance on the suicide classification tasks.

9.3 Experimental observations

A diverse set of experiments was conducted to answer various aspects of the main research question:

> *Can relevant and severe messages about suicide be automatically detected in Dutch user-generated content, and if so, which information sources and techniques contribute to classification performance?*

Chapters 7 and 8 provided a detailed overview of all the experimental results, which lead to the following observations.

Relevance detection

We find that a text classification approach is a viable and promising strategy for detecting social media posts that are about suicide. On the development data, all classifiers outperform the keyword baseline. The best-performing model, obtained after joint optimization with stratified feature groups, achieves an F_1 score of 92.69%, and offers a good balance between precision and recall. When the proposed system is scaled to large datasets with high class skew, it retains very high precision: false positives are virtually absent. The system is generally conservative in its predictions. All false negatives lack explicit mentions of suicide, suggesting that in order to improve recall, more implicit references need to be detected.

Severity detection

Posts that contain a severe threat of suicide are more complex to detect, both for human annotators and machine learning models. The scores are significantly below those for the relevance task, which is unsurprising given the task ambiguity and the smaller amount of training material. The best F_1 score of 69.51% is obtained with stratified feature group selection. The system finds 2 out of 3 severe posts, and only 1 in 4 suggested posts is not severe. From a usability perspective, this is very reasonable in terms of noise, and can be considered a step forward in automated prevention practice. Nevertheless, better recall is desirable. With cascaded classification, TiMBL is capable of achieving high recall (80%, i.e. 4 out of 5), although it does so at the expense of precision (50%). For applications where added noise is not problematic, this may be the preferred system.

The qualitative analysis on the scaling dataset reveals that the severity models are most successful in detecting posts in which an author personally discloses suicide ideation, especially when this is done in explicit terms. Posts about a third person are often incorrectly dismissed as relevant but insevere. More false positives are produced on this big dataset than with the relevance system, although more than half of them contain suicide risk factors, and are therefore not entirely irrelevant. At less than 0.01% of the data, noise is still acceptably low for the system to be usable in a real-world application.

Information sources

We defined a variety of features with the aim of gaining an insight into what kind of information is relevant for suicidality modelling. Overall, we find that virtually all feature groups are informative to some extent. More specifically, the following observations could be made:

- Both token and character bag-of-words features are often selected. We notice that ngrams based on the original words are mutually interchangeable with those based on lemmas. For the relevance task, token unigrams and bigrams are preferred, whereas for severity, there is a clear preference for longer ngrams: trigrams are selected, unigrams are discarded. This would indicate that relevant posts can be successfully identified with short keywords, whereas the added specificity of collocations is required for severity detection.
- Term features with non-exact matching are always included. This validates the approach of extracting highly salient collocations from a specialized corpus. Relaxed term matching also provides better abstraction than the token ngram or TERM-exact features.
- The abstraction obtained by clustering semantically related concepts into topics is beneficial. LSA features are found to perform very well, particularly for severity. Features with high amounts of topics are favoured, indicating that high topic granularity is most adequate to detect signals of suicidality.
- The assumption that negative (or lack of positive) polarity is associated with posts about suicide is confirmed. Features from the polarity lexicons are selected for both tasks. Additionally, we find that the polarity of the final words in a message is most informative.
- The miscellaneous feature groups are selected least often. For the severity task, named entity information is salient. We speculate that these features may help in labeling informative and journalistic messages as non-severe.

Model optimization

Optimization, as argued in Hoste (2005), is an essential exploration of the space of possible experiments, and allows reliable conclusions to be drawn about the performance of a machine learning method. We optimized the selected features and hyperparameters for our models with a genetic algorithm approach, which was found to be effective: optimization invariably improved performance, with

error reductions on the development set of up to 25% for both tasks. Optimized models also performed better on the development set, although less consistently so. We can conclude that optimization typically results in more robust models.

We optimized towards two fitness objectives: F_1, and F_2 for improved recall. It is of note that for both tasks, optimization towards F_2 often yields the best overall F_1 and F_2 score. We hypothesize that optimizing towards recall is the better strategy for this task. All classifiers optimized for F_1 obtain a score that is balanced in terms of precision and recall, whereas F_2 classifiers consistently achieve lower precision and higher recall. In other words, the different optimization objectives reliably steer the GA in the preferred direction, but the aim for better recall eventually leads to the best F_1 scores as well. It is plausible that F_1 optimization discards sub-optimal solutions with high recall before they can be fine-tuned for better precision.

Hyperparameter and joint optimization lead to better performance for the relevance task, especially in terms of recall. For severity, hyperparameters have little to no positive impact, and including them for optimization can even deteriorate the optimal results. We found this to be caused by search space sparsity, which can be remedied by changing the Gallop settings.

Unlike hyperparameter optimization, feature selection is always effective. Of the three tested strategies for feature selection, stratified feature group selection performed best. It offers more granularity by splitting large feature groups into ranked bins. The selection results demonstrate that this is beneficial: in the ngram feature groups, for example, more than half of the bins is removed. Not only does this result in better scores, it also makes for a model that requires fewer features. Furthermore, we find that strata are selected from all stratified feature groups. Instead of having to include or exclude entire groups, the search algorithm can pick their most useful subsets.

Cascaded classification

Cascades were proposed as an alternative to direct classification, with the potential advantage that models in a cascade derive simpler and more robust hypotheses. The experiments with cascades revealed that in our setup, they do not provide a performance benefit over direct classification in terms of F_1-score. It does, however, allow the use of TiMBL, which performs better than the direct and cascaded SVM systems in terms of recall and F_2. Future work should elucidate whether cascaded classification performance can be improved by reducing false negative errors percolating from the relevance filter.

Normalization

The application of text normalization has demonstrated its benefits for suicidality classification. We observe a dimensionality reduction in the full feature vectors of around 20%, and feature informativeness goes up. In the relevance classification experiments, the models with normalized vectors perform markedly better than the original ones: precision is slightly lower, but the improvement in recall constitutes an error reduction of almost 30%. For the relevance classifier as a first step in a cascaded severity model, a model using normalized vectors would likely be a superior candidate, although this is yet to be confirmed experimentally.

For direct severity classification, the normalized model performs better than the best original model, both in terms of precision and recall. It achieves the overall highest F_1 score on the development set of 71.91%, an error reduction of 9%. On the testset, precision is considerably lower. It seems that systems optimized to be less conservative in their predictions score well on the development set, but overgenerate on the held-out test data. We believe this effect may be partly explained by the small size of the test sample, which exaggerates the weight of idiosyncracies in the small number of positive instances. Testing on a sample with more positive material would clarify this.

We can conclude that normalization is a worthwhile effort: as an extrinsic evaluation, these experiments have demonstrated its utility for suicide detection, and more generally, text classification on user-generated content.

9.4 Future work

We are convinced that a classification-based approach to suicidality detection is promising, and that it can be instrumental in large-scale online suicide prevention efforts. The resources and systems developed in this study can serve as a benchmark for new approaches. We propose that future work be focused on the following areas.

Data

In Chapter 3, a number of drawbacks of the collected data were formulated: the amount of (severe) suicide-related posts available for training is limited, and because collection was mostly done based on search terms, posts that referenced suicide less explicitly were underrepresented in the corpus. A potential solution

could be to integrate the currently available detection system in a practical application and collect additional material. With active learning (Cohn et al. 1994, Olsson 2009), expert feedback could be used to selectively annotate unlabeled data, and refine the model. Such a real-world application would also allow usability testing. Another interesting semi-supervised approach is positive-versus-unknown classification (Liu et al. 2003, Yu et al. 2004), in which a classifier is provided with a small labeled set of examples (e.g. implicit suicide-related posts) and a large unlabeled set. It iteratively increases the set of negative instances, without misclassifying the positive ones. The remaining instances could be good candidates for annotation.

The current corpora contain forum and blog posts. Data collected from Twitter and Facebook would be valuable for testing detection performance on shorter content. Additional metadata, such as user information, post history and frequency, would also be very informative. The current detection approach considers each post in isolation. Profile-based modelling may be key to overcoming the limited scope a single message offers.

Ideally, social media content would be obtained from users that are known to suffer from suicide ideation. Currently, we annotate data from the perspective of the prevention worker, without knowledge about the actual context in which a message was written. Content from victims may allow more effective detection and prevention, but its collection creates practical and ethical issues that should be addressed.

Information sources

In this study, we considered a range of information sources for suicidality detection, but a wide variety of potentially informative features remains to be tested: tf-idf-weighted bag-of-words features, finer-grained topic models, the presence of code switching (e.g. from Dutch to English), risk factors or 'allness' terms (e.g. *everyone*, *never*, *completely*), relative part-of-speech frequencies, average sentence length, readability scores, etc.

Recall

In our experiments, the aim was to obtain models that gave equal importance to precision and recall. Emphasizing recall at the expense of precision could render a system practically unusable if the user is flooded with false positives. It appeared, however, that the experimental systems are relatively precise, so it might be worthwhile to investigate solutions that tolerate more false positives

and retrieve more positive instances that are currently missed. We see a number of opportunities to achieve this objective.

Models can be optimized towards higher recall, e.g. by using $F_{\beta=4}$ or $F_{\beta=8}$ as a fitness function. Experiments with instance selection and data sampling to remove class skew in the training data could also benefit recall. For feature ranking and filtering, the bi-normal separation metric proposed by Forman (2003) could be superior to information gain. It was shown to compare very favorably to other metrics in high-skew situations, and when recall is the objective. We believe that a cascaded classification approach with a series of high-recall, low-precision filters (as proposed by Viola and Jones (2001)) should be investigated. Lastly, further research on automatic normalization, which was shown to improve lexical and overall recall, could be beneficial.

APPENDIX A

Publications

This appendix contains a list of all peer-reviewed journal and conference proceedings publications from the period 2010-2014.

- 2014
  - Van de Kauter, M., Desmet, B. and Hoste, V. *The Good, the Bad and the Implicit: A Comprehensive Approach to Annotating Explicit and Implicit Sentiment.* Language Resources and Evaluation. Accepted for publication.
  - De Clercq, O., Hoste, V., Desmet, B., van Oosten, P., De Cock, M. and Macken, L. *Using the Crowd for Readability Prediction.* Natural Language Engineering, 20 (3), 293-235. Cambridge Journals Online.
  - De Clercq, O., Schulz, S., Desmet, B. and Hoste, V. *Towards Shared Datasets for Normalization Research.* In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk and Stelios Piperidis (eds.), Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), 1218-1223. European Language Resources Association (ELRA), Reykjavik, Iceland.

  - Desmet, B. and Hoste, V. *Recognising suicidal messages in Dutch social media.* In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk and Stelios Piperidis (eds.), Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), 830-835. European Language Resources Association (ELRA), Reykjavik, Iceland.

- 2013

  - De Clercq, O., Schulz, S., Desmet, B., Lefever, E. and Hoste, V. *Normalization of Dutch User-Generated Content.* Proceedings of the 9th International Conference on Recent Advances in Natural Language Processing (RANLP 2013). Hissar, Bulgaria.
  - Desmet, B. and Hoste, V. *Fine-Grained Dutch Named Entity Recognition.* Language Resources and Evaluation, 48 (2), 307-343. Springer Netherlands.
  - Desmet, B. and Hoste, V. *Emotion Detection in Suicide Notes.* Expert Systems with Applications, 40 (16), 6351-6358.
  - Van de Kauter, M., Coorman, G., Lefever, E., Desmet, B., Macken, L. and Hoste, V. *LeTs Preprocess: The multilingual LT3 linguistic preprocessing toolkit.* Computational Linguistics in the Netherlands Journal, 3, 103-120.

- 2012

  - Desmet, B. and Hoste, V. *Combining Lexico-semantic Features for Emotion Classification in Suicide Notes.* Biomedical Informatics Insights, 5, 125-128. Libertas Academica.

- 2011

  - Desmet, B. and Hoste, V. *Using Classifier Ensembles for Named Entity Recognition in Dutch.* In P. De Causmaecker, J. Maervoet, T. Messelis, K. Verbeeck, and T. Vermeulen (eds.), Proceedings of the 23rd Benelux Conference on Artificial Intelligence. Ghent, Belgium.

- 2010
  - Desmet, B. and Hoste, V. *Dutch Named Entity Recognition using Classifier Ensembles.* In T. Markus, P. Monachesi, and E. Westerhout (eds.), Computational Linguistics in the Netherlands 2010: selected papers from the twentieth CLIN meeting. Netherlands Graduate School of Linguistics, Utrecht, Netherlands.
  - Desmet, B. and Hoste, V. *Towards a Balanced Named Entity Corpus for Dutch.* In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias (eds.), Proceedings of the seventh International Conference on Language Resources and Evaluation (LREC'10). European Language Resources Association, Valletta, Malta.
  - Vanopstal, K., Desmet, B. and Hoste, V. *Towards a Learning Approach for Abbreviation Detection and Resolution.* In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias (eds.), Proceedings of the seventh International Conference on Language Resources and Evaluation (LREC'10). European Language Resources Association, Valletta, Malta.

APPENDIX B

Multiword suicide lexicon

List of 251 multiword terms extracted from the CPZ chat transcript corpus. All lexicon entries have been lemmatized with Pattern, terms contain only content words.

aantal jaar	aantal zaak	algemeen welzijn
algemeen welzijnswerk	begin gesprek	beslissing zelfmoord
best oplossing	best vriend	best vriendin
beter beeld	beter gevoel	blauw plek
coming out	concreet plan	constant ruzie
crisis dienst	dankje babbel	depressief gevoel
diepst gevoelen	dik knuffel	dom ding
donker gedacht	drang snijd	drastisch beslissing
dreigend miskraam	dubbel gevoel	duwtje rug
dwingen opname	echt deugden	echt hulp
echt leven	echt nie	echt nood
echt pijn	echt reden	echt steun

echt super
echt vriend
echt zelfmoord
eigenlijk zelfmoord
einde raad
emotioneel pijn
enorm stap
erge pijn
fijn avond
flauw idee
fysiek pijn
gedacht zelfmoord
gek ding
gespecialiseren mens
gesprek stilletje
geven moment
gevoel eenzaamheid
gevoel machteloosheid
gevoel rust
gevoelen pijn
gevoelen verdriet
gezellig avond
glas wijn
goed avond
goed band
goed begrijpen
goed contact
goed ding
goed eigenschap
goed gesprek
goed gevoel
goed idee
goed inkomen
goed manier
goed moeder
goed nieuws
goed raad
goed relatie
goed richting
goed vriend
goed vriendin
goei avond
goei vriendin
groot lijn
groot stap
grootst nood
grootst probleem
half jaar
half uur
hecht band
heel avond
heel boterham
heel dag
heel dapper
heel goed idee
heel groot stap
heel jaar
heel klein beetje
heel laag zelfbeeld
heel leven
heel moeilijk moment
heel stap
heel stuk
heel tijd
heel verhaal
heel week
heel wereld
huidig vriend
immen verdriet
impulsief bui
innen gevang
innerlijk pijn
jaar dood
jong leeftijd
jong meisje
jongeren advies centrum
jongeren adviescentrum
jongst broer
juist beslissing
juist hulp
juist oplossing
keer keer
keer maand
keuze zelfmoord
klein beetje
klein ding
klein kind
klein stapje
komennen dag
komennen week
laag zelfbeeld
laatst dag
laatst gesprek
laatst jaar
laatst keer
laatst maand
laatst moment
laatst poging
laatst stap
laatst tijd
laatst week
lager school
lang gesprek
lang periode
lang termijn
lang tijd
lang verhaal
lang weg
langer tijd
lastig val
leuk ding

leven dood
luisteren oor
mens pijn
mens zelfmoordgedacht
moeilijk moment
moeilijk stap
mogelijk oplossing
mooi toekomst
negatief gevoelen
nieuw poging
nieuw vriendin
ongeboren kind
open gesprek
overdosis medicatie
overlijden zanger
paar jaar
paar week
pijnlijk gevoelen
professioneel hulp
psychisch pijn
rustig avond
rustiger leven
slecht gevoel
sociaal contact
soort ding
sterk gevoelen
vervelend gevoel
volgennen keer
vorig keer
vorig week
vrij loop
weg chat
zin leven
zwaar ding
zwart gat
lichamelijk pijn
maatschappelijk werker
mens probleem
mn leven
moeilijk periode
moeilijk vinden
moment controle
nabestaand zelfdoding
nieuw baby
nieuw relatie
normaal leven
ongedaan maken
oudst zoon
overdosis medicijn
paar dag
paar keer
paaz afdeling
plan zelfmoord
professioneel hulpverlener
raad vragen
rustig manier
slecht ding
slecht gevoelen
sociaal dienst
sprankeltje hoop
teel onthaal
volgen jaar
volgennen week
vorig maand
vreselijk situatie
vrouwelijk psychiater
weg huis
zoek rust
zwaar kruis
zwart gedacht
licht vorm
mens hoogte
mens verdriet
moeilijk karakter
moeilijk situatie
moeilijk vraag
moment zelfmoord
nabij toekomst
nieuw begin
nieuw vriend
nummer zelfmoordlijn
onmiddellijk omgeving
oudst zus
overdosis pil
paar ding
paar maand
papa bed
positief ding
psychisch hulp
raar vraag
rustig nacht
slecht gesprek
slecht idee
sociaal zelfmoord
sterk band
telefonisch contact
volgennen afspraak
vorig jaar
vorig poging
vriendin zelfmoord
waardig manier
zelfmoordlijn contact
zorg kind
zwaar last

List of Figures

1.1 Schematic overview of the suicidal process. 3

1.2 Popular Twitter post following the death of actor Robin Williams, who voiced the genie character in Disney's *Alladin*. 6

3.1 Schematic overview of the text level annotations. Round radio buttons indicate exclusive choices, square checkboxes indicate non-exclusive options. 31

3.2 Main view of the BRAT annotation interface, with visible menubar. 38

3.3 Panel for structured text level annotation in BRAT. 39

3.4 Panel for text span annotation in BRAT. 39

3.5 Breakdown of annotations per category. 44

4.1 Singular value decomposition of the $m \times n$ matrix X with rank r. 59

5.1 Possible classification hypotheses as decision boundaries in a two-dimensional feature vector space. 67

5.2 Support vectors and maximized margin. 67

5.3 Examples of k Nearest Neighbour binary classification in a two-dimensional feature vector space. The star represents an unseen instance to be classified. 71

6.1 A potential solution to the problem of joint optimization in TiMBL, coded as a genotype. Its structure is dictated by the genome, the genetic representation of the search space. 91

6.2 Graphical representation of an optimization run with a genetic algorithm. 92

6.3 Example of mutation applied to an individual, where 4 genes have been randomly changed. 93

6.4 Example of single point crossover, where the genotypes of two parents are combined to produce two children. 94

7.1 Histograms of the post length distribution in the two constituent parts of the development corpus: the 1 000 post sample from the suicide corpus on the left, and the 9 000 post stratified sample from the reference corpus on the right. 103

7.2 Histograms of the post length distribution in four stratified samples from the reference corpus: the held-out test set (top left), and the three scaling sets of size 20 000, 70 000 and 200 000. . . . 105

7.3 Absolute scores (top row) and percentage-wise error reduction after optimization (bottom row), for the relevance task on the development set. Scores are expressed in terms of F_1 (left) or F_2 (right), depending on the optimization objective. 110

7.4 Absolute scores (top row) and percentage-wise error reduction after optimization (bottom row), for the severity task on the development set. Scores are expressed in terms of F_1 (left) or F_2 (right), depending on the optimization objective. 122

7.5 Number of true and false positives on the held-out and scaling datasets, using the best F_1 and F_2 classifiers for the relevance and severity tasks. False positives are divided into two groups, depending on whether they contain risk factors. 138

8.1 Visualization of the WER reduction on the SMS data set using seven different setups. 152

8.2 Visualization of BLEU on the SMS data set using seven different setups. 153

8.3 Proportion of missing normalization per error category. 155

List of Tables

3.1 Inter-annotator agreement scores for the relevance and severity tasks. 40

3.2 Post length in the Netlog suicide corpus (mean, standard deviation, minimum, maximum), in terms of lines, tokens and characters. 43

4.1 Pattern tokenization and lemmatization output for an example sentence before and after cleaning and pretokenization. 50

4.2 Overview of the feature groups. Features are based on the *original*, *clean* or *last* text layer, and can be integers, real-valued or binary. 62

5.1 Confusion matrix for a binary classification task. 75

6.1 Feature group sizes before and after applying information gain filtering for both classification tasks. Features with an IG value below 0.001 were removed. 86

7.1 Two examples of the k-nearest fit individuals, with different values for precision (to n significant figures) and k. 107

7.2 Baseline scores on the development and held-out datasets, for the relevance and severity tasks. 107

7.3 Relevance classification scores on the development set, optimized towards F_1 (HO = hyperparameter optimization, FS = feature selection). 109

7.4 Relevance classification scores on the development set, optimized towards F_2 (HO = hyperparameter optimization, FS = feature selection). 109

7.5 Relevance classification scores on the held-out testset, optimized towards F_1 (HO = hyperparameter optimization, FS = feature selection). 113

7.6 Relevance classification scores on the held-out testset, optimized towards F_2 (HO = hyperparameter optimization, FS = feature selection). 113

7.7 Selected hyperparameters for the relevance task. 114

7.8 Feature group selection status in all relevance models with regular or nbest feature group selection (FS), with or without hyperparameter optimization (HO), and optimized towards F_1 or F_2. Cell colour indicates the relative frequency of selection (darker = more often selected). 115

7.9 Selection status of all feature group strata, in the relevance model with joint optimization towards F_2. Cell colour indicates the relative frequency of selection (darker = more often selected). . . 118

7.10 Direct severity classification scores on the development set, optimized towards F_1 (HO = hyperparameter optimization, FS = feature selection). 121

7.11 Direct severity classification scores on the development set, optimized towards F_2 (HO = hyperparameter optimization, FS = feature selection). 121

7.12 Direct severity classification scores on the held-out testset, optimized towards F_1 (HO = hyperparameter optimization, FS = feature selection). 124

7.13 Direct severity classification scores on the held-out testset, optimized towards F_2 (HO = hyperparameter optimization, FS = feature selection). 124

7.14 Selected hyperparameters for the direct severity task. 125

7.15 Feature group selection status in all severity models with regular or nbest feature group selection (FS), with or without hyperparameter optimization (HO), and optimized towards F_1 or F_2. Cell colour indicates the relative frequency of selection (darker = more often selected). 126

7.16 Selection status of all feature group strata, in the severity model without hyperparameter optimization, optimized towards F_2. Cell colour indicates the relative frequency of selection (darker = more often selected). 128

7.17 Cascaded severity classification scores on the development set, with gold standard relevance labels. 130

7.18 Selected hyperparameters for the cascaded severity task with gold standard relevance labels. 132

7.19 Feature group selection status in cascaded severity models with regular or nbest feature group selection (FS), with or without hyperparameter optimization (HO), and using gold standard relevance labels. Cell colour indicates the relative frequency of selection (darker = more often selected). 133

7.20 Cascaded severity classification scores on the development set, with automatically predicted relevance labels. 135

7.21 Cascaded severity classification scores on the held-out testset, with automatically predicted relevance labels. 135

8.1 Examples from the three social media genres under study, representing the original utterance, its normalized version and an English translation. 147

8.2 Normalization statistics of the three UGC genres. On the left: message and token counts; on the right: operation counts. 148

8.3 WER and BLEU performance of the seven setups, trained on the SMS genre and tested on all genres. 151

8.4 Absolute and relative number of missed operations at the character level. 154

8.5 Examples of missing normalizations of each error category. 155

8.6 Example of a Netlog post before and after normalization. Tokens with correct normalization edits are boldfaced, missing or incorrect normalizations and unsolvable cases are italicized. . . . 159

8.7 Number of features, based on original or normalized text. 160

8.8 Comparison of results on the relevance task, with and without normalization. 161

8.9 Comparison of results on the severity task, with and without normalization. 162

Bibliography

Aha, D. and Bankert, R.: 1996, A comparative evaluation of sequential feature selection algorithms, *in* D. Fischer and J.-H. Lenz (eds), *Artificial intelligence and statistics V*, New York: Springer Verlag.

Aha, D., Kibler, D. and Albert, M.: 1991, Instance-based learning algorithms, *Machine Learning* **6**, 37–66.

Allison, B. N. and Schultz, J. B.: 2001, Interpersonal identity formation during early adolescence, *Adolescence* **36**(143), 509–23.

Alonzo, M. and Aiken, M.: 2004, Flaming in electronic communication, *Decision Support Systems* **36**(3), 205–213.

Androutsopoulos, J.: 2007, Neue Medien neue Schriftlichkeit?, *Mitteilungen des Deutschen Germanistenverbandes* **1**(7), 72–97.

Aw, A., Min, Z., Juan, X. and Jian, S.: 2006, A Phrase-based Statistical Model for SMS Text Normalization, *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, Sydney, Australia, pp. 33–40.

Baayen, H. R., Piepenbrock, R. and Gulikers, L.: 1995, The CELEX Lexical Database. Linguistic Data Consortium, CD-ROM.

Baron, N. S.: 2003, Language of the internet, *The Stanford Handbook for Language Engineers* pp. 59–127.

Baroni, M. and Bernardini, S.: 2004, BootCaT : Bootstrapping Corpora and Terms from the Web, *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC '04)*, Lisbon, Portugal, pp. 1313–1316.

Beaufort, R., Roekhaut, S., Cougnon, L.-A. and Fairon, C.: 2010, A hybrid rule/model-based finite-state framework for normalizing sms messages, *Proceedings of ACL*, pp. 770–779.

Berger, A., Caruana, R., Cohn, D., Freitag, D. and Mittal, V.: 2000, Bridging the Lexical Chasm: Statistical Approaches to Answer Finding, *Proc. Int. Conf. Research and Development in Information Retrieval*, pp. 192–199.

Black, S. T.: 1993, Comparing genuine and simulated suicide notes: a new perspective., *Journal of Consulting and Clinical Psychology* **61**(4), 699–702.

Burke, S. M.: 2001, Unidecode!, *Sys Admin* **10**(12), 54–60.

Cappadocia, M. C., Craig, W. M. and Pepler, D.: 2013, Cyberbullying: Prevalence, Stability, and Risk Factors during Adolescence, *Canadian Journal of School Psychology* **28**(2), 171–192.

Carletta, J.: 1996, Assessing Agreement on Classification Tasks: The Kappa Statistic, *Computational Linguistics* **22**(2), 249–254.

Chang, C.-C. and Lin, C.-J.: 2011, LIBSVM: A library for support vector machines, *ACM Transactions on Intelligent Systems and Technology* **2**(3), 1–27.

Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S. and Basu, A.: 2007, Investigating and modeling the structure of texting language, *International Journal on Document Analysis and Recognition* **10**(3), 157–174.

Clement, R. and Sharp, D.: 2003, Ngram and Bayesian Classification of Documents for Topic and Authorship, *Literary and Linguistic Computing* **18**(4), 423–447.

Cohn, D., Atlas, L. and Ladner, R.: 1994, Improving generalization with active learning, *Machine Learning* **15**(2), 201–221.

Cook, P. and Stevenson, S.: 2009, An unsupervised model for text message normalization, *Proceedings of the NAACL HLT Workshop on Computational Approaches to Linguistic Creativity*.

Cover, T. M. and Thomas, J. A.: 2012, *Elements of Information Theory*, John Wiley & Sons, New York.

Cristianini, N. and Shawe-Taylor, J.: 2000, *An Introduction to Support Vector Machines and other kernel-based learning methods*, Cambridge University Press.

Dadvar, M.: 2014, *Experts and machines united against cyberbullying*, PhD thesis, University of Twente.

Daelemans, W., van den Bosch, A. and Zavrel, J.: 1999, Forgetting Exceptions is Harmful in Language Learning, *Machine Learning* **34**(1-3), 11–41.

Daelemans, W., Zavrel, J., van der Sloot, K. and van den Bosch, A.: 2009, TiMBL: Tilburg Memory Based Learner, version 6.2, Reference Guide, *Technical Report 09-01*, ILK Research Group.

Dagan, I. and Church, K.: 1994, Termight: identifying and translating technical terminology, *Proceedings of Applied Language Processing*, pp. 34–40.

Daille, B.: 1996, Study and implementation of combined techniques for automatic extraction of terminology, *in* J. L. Klavans and P. Resnik (eds), *The balancing act: combining symbolic and statistical aproaches to language*, MIT Press, Massachusetts.

De Clercq, O., Schulz, S., Desmet, B. and Hoste, V.: 2014, Towards Shared Datasets for Normalization Research, *in* N. Calzolari, K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk and P. Stelios (eds), *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, European Language Resources Association (ELRA), Reykjavik, Iceland, pp. 1218–1223.

De Clercq, O., Schulz, S., Desmet, B., Lefever, E. and Hoste, V.: 2013, Normalization of Dutch User-Generated Content, *Proceedings of the 9th International Conference on Recent Advances in Natural Language Processing (RANLP 2013)*, pp. 179–188.

De Jaegere, E., Portzky, G., van den Berg, M. and Wallyn, S.: 2013, Ethical Guidelines for Technology-Based Suicide Prevention Programmes, *Technical report*.

De Jaegere, E., Vancayseele, N., Portzky, G. and Van Heeringen, C.: 2012, De Epidemiologie van Suïcidepogingen in Vlaanderen, *Technical report*, Unit for Suicide Research, Ghent University.

De Leo, D., Burgis, S., Bertolote, J. M., Kerkhof, A. J. F. M. and Bille-Brahe, U.: 2006, Definitions of Suicidal Behavior, *Crisis: The Journal of Crisis Intervention and Suicide Prevention* **27**(1), 4–15.

De Smedt, T. and Daelemans, W.: 2012a, Pattern for Python, *Journal of Machine Learning Research* **13**, 2063–2067.

De Smedt, T. and Daelemans, W.: 2012b, Vreselijk mooi! (terribly beautiful): A Subjectivity Lexicon for Dutch Adjectives, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, pp. 3568–3572.

Demberg, V., Schmid, H. and Möhler, G.: 2007, Phonological Constraints and Morphological Preprocessing for Grapheme-to-Phoneme Conversion, *45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, Association for Computational Linguistics, Prague, Czech Republic.

Desmet, B. and Hoste, V.: 2013a, Emotion Detection in Suicide Notes, *Expert Systems with Applications* **40**(16), 6351–6358.

Desmet, B. and Hoste, V.: 2013b, Fine-grained Dutch named entity recognition, *Language Resources and Evaluation* **48**(2), 307–343.

Desmet, B. and Hoste, V.: 2014, Recognising suicidal messages in Dutch social media, *in* N. Calzolari, K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk and P. Stelios (eds), *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, European Language Resources Association (ELRA), Reykjavik, Iceland, pp. 830–835.

Dewey, C.: 2014, Suicide contagion and social media: The dangers of sharing 'Genie, you're free'.
URL: *http://www.washingtonpost.com/news/the-intersect/wp/2014/08/12/suicide-contagion-and-social-media-the-dangers-of-sharing-genie-youre-free/*

Dinarelli, M. and Rosset, S.: 2011, Models Cascade for Tree-Structured Named Entity Detection, *Proceedings of the 5th International Joint Conference on Natural Language Processing*, Chiang Mai, Thailand, pp. 1269–1278.

Drouin, P.: 2003, Term extraction using non-technical corpora as a point of leverage, *Terminology* **9**(1), 99–116.

Dudani, S.: 1976, The Distance-Weighted k-Nearest-Neighbor Rule, *IEEE Transactions on Systems, Man and Cybernetics* **6**(4), 325–327.

Edelman, A. M. and Renshaw, S. L.: 1982, Genuine versus simulated suicide notes: An issue revisited through discourse analysis, *Suicide and Life-Threatening Behavior* **12**(2), 103–113.

Eggermont, S., Van den Bulck, J. and Kerkhof, A.: 2007, Het Werthereffect: de rol van media in suïcidaal gedrag, *Handboek Suïcidaal Gedrag*, De Tijdstroom, Utrecht, pp. 315–323.

Esuli, A. and Sebastiani, F.: 2006, Sentiwordnet: a publicly available lexical resource for opinion mining, *Proceedings of LREC 2006*.

Forman, G.: 2003, An Extensive Empirical Study of Feature Selection Metrics for Text Classification, *Journal of Machine Learning Research* **3**, 1289–1305.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M. and Gagné, C.: 2012, DEAP: Evolutionary Algorithms Made Easy, *Journal of Machine Learning Research* **13**, 2171–2175.

Foster, J., Cetinoglu, O., Wagner, J., Roux, J. L., Hogan, S., Nivre, J., Hogan, D. and van Genabith, J.: 2011, #hard-toparse: POS tagging and parsing the twitterverse, *Proceedings of the AAAI workshop on Analyzing Microtext*, pp. 20–25.

Frantzi, K. and Ananiadou, S.: 1999, The C-value / NC-value domain independent method for multi-word term extraction, *Journal of Natural Language Processing* **6**(3), 145–179.

Gisle, L.: 2008, Mentale Gezondheid, *Technical report*, Wetenschappelijk Instituut Volksgezondheid, Brussel.

Goldberg, D.: 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley.

Goldberg, D. and Deb, K.: 1991, A Comparative Analysis of Selection Schemes Used in Genetic Algorithms, *Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, California, USA, pp. 69–93.

Goldberg, Y. and Elhadad, M.: 2008, splitSVM: Fast, Space-Efficient, non-Heuristic, Polynomial Kernel Computation for NLP Applications, *Proceedings of ACL-08: HLT, Short Papers*, number June, Association for Computational Linguistics, Columbus, US, pp. 237–240.

Gottschalk, L. A. and Gleser, G. C.: 1960, An analysis of the verbal content of suicide notes, *British Journal of Medical Psychology* **33**(3), 195–204.

Haerian, K., Salmasian, H. and Friedman, C.: 2012, Methods for Identifying Suicide or Suicidal Ideation in EHRs, *AMIA Annual Symposium Proceedings*, pp. 1244–1253.

Harrell, F. E.: 2001, *Regression Modeling Strategies*, Vol. 64, Springer.

Hawton, K. and van Heeringen, C. (eds): 2000, *The international handbook of suicide and attempted suicide*, Wiley, Chichester.

Hinduja, S. and Patchin, J. W.: 2010, Bullying, cyberbullying, and suicide, *Archives of Suicide Research* **14**(3), 206–221.

Holland, J.: 1975, *Adaptation in natural and artificial Systems*, MIT Press.

Hoste, V.: 2005, *Optimization Issues in Machine Learning of Coreference Resolution*, PhD thesis, Antwerp University.

Hoste, V., Hendrickx, I., Daelemans, W. and van den Bosch, A.: 2002, Parameter Optimization for Machine-Learning of Word Sense Disambiguation, *Natural Language Engineering, Special Issue on Word Sense Disambiguation Systems* **8**, 311–325.

Hsu, C.-w., Chang, C.-c. and Lin, C.-j.: 2010, A Practical Guide to Support Vector Classification, **1**(1), 1–16.

Huang, Y.-P., Goh, T. and Liew, C. L.: 2007, Hunting Suicide Notes in Web 2.0 - Preliminary Findings, *Ninth IEEE International Symposium on Multimedia Workshops (ISMW 2007)*, IEEE, pp. 517–521.

Jashinsky, J., Burton, S. H., Hanson, C. L., West, J., Giraud-Carrier, C., Barnes, M. D. and Argyle, T.: 2013, Tracking Suicide Risk Factors Through Twitter in the US., *Crisis* pp. 1–9.

Jijkoun, V. and Hofmann, K.: 2009, Generating a non-English subjectivity lexicon: relations that matter, *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 398–405.

Joachims, T.: 1998, Text Categorization with Support Vector Machines: Learning with Many Relevant Features, *Proceedings of the European Conference on Machine Learning*, Springer.

Jurafsky, D. and Martin, J. H.: 2009, *Speech and Language Processing: an introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall.

Kerkhof, A.: 2012, Calculating the burden of disease of suicide, attempted suicide, and suicide ideation by estimating disability weights, *Crisis* **33**(2), 63–65.

Kerkhof, A. and Van Heeringen, C.: 2000, Richtlijnen voor de behandeling van suïcidaliteit, *in* A. Kerkhof and C. Van Heeringen (eds), *Behandelingsstrategieën bij suïcidaliteit*, Bohn Stafleu Van Loghum, Houten, pp. 148–159.

Kerkhof, A. and van Luyn, J.: 2010, *Suïcidepreventie in de praktijk*, Bohn Stafleu van Loghum, Houten.

Kestemont, M., Peersman, C., Decker, B. D., Pauw, G. D., Luyckx, K., Morante, R., Vaassen, F., van de Loo, J. and Daelemans, W.: 2012, The netlog corpus. a resource for the study of flemish dutch internet language, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey.

Kirkpatrick, S., Gelatt Jr., C. D. and Vecchi, M. P.: 1983, Optimization by Simulated Annealing, *Science* **220**(4598), 671–680.

Kobus, C., François, Y. and Géraldine, D.: 2008, Normalizing SMS: are two metaphors better than one?, *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, Manchester, UK, pp. 441–448.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A. and Herbst, E.: 2007, Moses: Open Source Toolkit for Statistical Machine Translation, *Proceedings of the ACL 2007 Demo and Poster Sessions*, Prague, Czech Republic, pp. 177–180.

Kökciyan, N., Çelebi, A., Özgür, A. and Üsküdarli, S.: 2013, BOUNCE: Sentiment Classification in Twitter using Rich Feature Sets, *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, Atlanta, Georgia, USA, pp. 554–561.

Kolak, O., Byrne, W. J. and Resnik, P.: 2003, A Generative Probabilistic OCR Model for NLP Applications, *HLT-NAACL 2003: conference combining Human Language Technology conference series and the North American Chapter of the Association for Computational Linguistics conference series*, Edmonton, Canada.

Kübler, S. and Mohamed, E.: 2008, Memory-based vocalization of Arabic, *Proceedings of the LREC Workshop on HLT and NLP within the Arabic World.*, Marrakech, Morocco.

Landauer, T., Foltz, P. and Laham, D.: 1998, An introduction to latent semantic analysis, *Discourse processes* **25**, 259–284.

Landauer, T. K. and Dutnais, S. T.: 1997, A Solution to Plato's Problem: the Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge, *Psychological Review* **104**(2), 211–240.

Leenaars, A. A.: 1988, *Suicide notes: Predictive clues and patterns*, Human Sciences Press, New York.

Liu, B.: 2010, Sentiment Analysis and Subjectivity, *in* N. Indurkhya and F. J. Damerau (eds), *Handbook of Natural Language Processing, Second Edition*, Chapman & Hall/CRC Press, Boca Raton, pp. 627–664.

Liu, B.: 2011, *Web Data Mining*, Springer Berlin / Heidelberg.

Liu, B., Dai, Y., Li, X., Lee, W. S. and Yu, P. S.: 2003, Building text classifiers using positive and unlabeled examples, *Proceedings of the Third IEEE International Conference on Data Mining (ICDM2003)*, IEEE, pp. 179–186.

Liu, F., Weng, F. and Jiang, X.: 2012, A broad-coverage normalization system for social media language, *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1035–1044.

Liu, F., Weng, F., Wang, B. and Liu, Y.: 2011, Insertion, deletion or substitution? Normalizing text messages without pre-categorization nor supervision, *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 71–76.

Liu, X., Shaodian Zhang, F. W. and Zhou, M.: 2011, Recognizing named entities in tweets, *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 359–367.

Lobo, F. G., Lima, C. F. and Michalewicz, Z.: 2007, *Parameter setting in evolutionary algorithms*, Springer, Berlin.

Lopez de Lacalle, O.: 2009, *Domain-Specific Word Sense Disambiguation*, PhD thesis.

Luxton, D. D., June, J. D. and Kinn, J. T.: 2011, Technology-based suicide prevention: current applications and future directions., *Telemedicine and e-Health* **17**(1), 50–54.

Macken, L., Lefever, E. and Hoste, V.: 2013, TExSIS: Bilingual Terminology Extraction from Parallel Corpora Using Chunk-based Alignment, *Terminology* **19**(1), 1–30.

Manning, C. D., Raghavan, P. and Schütze, H.: 2008, *Introduction to Information Retrieval*, Cambridge University Press. online version: http://nlp.stanford.edu/IR-book/html/htmledition/mybook.html.

Marzuk, P. M., Tardiff, K. and Leon, A. C.: 1994, Increase in fatal suicidal self-poisonings and suffocations in the year Final Exit was published, *American Journal of Psychiatry* **151**, 1813–1814.

Mathers, C. D., Boerma, T. and Ma Fat, D.: 2009, Global and regional causes of death, *British Medical Bulletin* **92**, 7–32.

Matykiewicz, P., Duch, W. and Pestian, J.: 2009, Clustering semantic spaces of suicide notes and newsgroups articles, (June), 179–184.

McGhee, I., Bayzick, J., Kontostathis, A., Edwards, L., McBride, A. and Jakubowski, E.: 2011, Learning to identify internet sexual predation, *International Journal of Ele* **15**(3), 103–122.

Mendes, P. N., Jakob, M., García-Silva, A. and Bizer, C.: 2011, DBpedia Spotlight : Shedding Light on the Web of Documents, *Proceedings of the 7th International Conference on Semantic Systems*, pp. 1–8.

Michalewicz, Z. and Schmidt, M.: 2007, Parameter Control in Practice, *Parameter setting in evolutionary algorithms*, Vol. 294, Springer Berlin Heidelberg, pp. 277–294.

Michelbacher, L., Kothari, A., Forst, M., Lioma, C. and Schütze, H.: 2011, A Cascaded Classification Approach to Semantic Head Recognition, *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Edinburgh, Scotland, pp. 793–803.

Minguillón Alfonso, J.: 2002, *On cascading small decision trees*, PhD thesis, Universitat Autònoma de Barcelona.

Miniño, A. M. and Murphy, S. L.: 2012, Death in the United States, 2010, *Technical Report 99*, National Center for Health Statistics.

Motto, J. A. and Bostrom, A. G.: 2001, A randomized controlled trial of postcrisis suicide prevention, *Psychiatric Services* **52**(6), 828–833.

Mulholland, M. and Quinn, J.: 2013, Suicidal Tendencies: The Automatic Classification of Suicidal and Non-Suicidal Lyricists Using NLP, *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, number October, Nagoya, Japan, pp. 680–684.

Munezero, M., Kakkonen, T. and Montero, C. S.: 2011, No Title, *Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2011)*, pp. 20–27.

Naert, S.: 2013, *Uitbreiden van de annotatietool brat*, Master's thesis.

Neeleman, J.: 2007, Epidemiologie van suïcidaal gedrag, *Handboek Suïcidaal Gedrag*, De Tijdstroom, Utrecht, chapter 2, pp. 31–48.

O'Carroll, P. W., Berman, A. L., Maris, R. W., Moscicki, E. K., Tanney, B. L. and Silverman, M. M.: 1996, Beyond the Tower of Babel: a nomenclature for suicidology., *Suicide and Life-Threatening Behavior* **26**(3), 237–252.

Ogilvie, D. M., Stone, P. J. and Shneidman, E. S.: 1966, Some characteristics of genuine versus simulated suicide notes, *in* P. J. Stone, D. C. Dunphy, M. S. Smith and D. M. Ogilvie (eds), *The General Inquirer: A Computer Approach to Content Analysis*, MIT Press, Cambridge, Massachusetts.

Olsson, F.: 2009, A literature survey of active machine learning in the context of natural language processing, *Technical Report T2009:06*.

Oostdijk, N.: 2000, The spoken dutch corpus. Outline and rst evaluation, *Proceedings of the Second International Conference on Language Resources and Evaluation*, pp. 887–894.

Oostdijk, N., Reynaert, M., Hoste, V. and Schuurman, I.: 2013, The Construction of a 500-Million-Word Reference Corpus of Contemporary Written Dutch, *Theory and Applications of Natural Language Processing* (Essential Speech and Language Technology for Dutch), 219–247.

Opitz, D. and Maclin, R.: 1999, Popular Ensemble Methods: an empirical study, *Journal of Artificial Intelligence Research* **11**, 169–198.

Osgood, C. E. and Walker, E. G.: 1959, Motivation and language behavior: a content analysis of suicide notes., *Journal of Abnormal and Social Psychology* **59**(1), 58–67.

Pantel, P. and Lin, D.: 2001, A Statistical Corpus-Based Term Extractor, *Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, pp. 36–46.

Peersman, C., Vaassen, F. and Asch, V. V.: 2012, Conversation Level Constraints on Pedophile Detection in Chat Rooms, *CLEF 2012 Conference and Labs of the Evaluation Forum - Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN)*, Rome, Italy.

Pendar, N.: 2007, Toward spotting the pedophile. Telling victim from predator in text chats., *Proceedings of the International Conference on Semantic Computing (ICSC)*.

Pennebaker, J. W., Francis, M. E. and Booth, R. J.: 2001, *Linguistic Inquiry and Word Count: LIWC 2001*, Lawrence Erlbaum Associates, Mahwah, NJ.

Pennell, D. L. and Liu, Y.: 2011, A character-level machine translation approach for normalization of SMS abbreviations, *Proceedings of 5th International Joint Conference on Natural Language Processing*, Asian Federation of Natural Language Processing, Chiang Mai, Thailand.

Pestian, J., Matykiewicz, P., Linn-Gust, M., South, B., Uzuner, O., Wiebe, J., Cohen, K. B., Hurdle, J. and Brew, C.: 2012, Sentiment Analysis of Suicide Notes: A Shared Task, *Biomedical Informatics Insights* **5**, 3–16.

Pestian, J., Nasrallah, H., Matykiewicz, P., Bennett, A. and Leenaars, A.: 2010, Suicide Note Classification Using Natural Language Processing : A Content Analysis, *Biomedical Informatics Insights* **3**, 19–28.

Pestian, J. P., Matykiewicz, P., Grupp-Phelan, J., Arszman Lavanier, S., Combs, J. and Kowatch, R.: 2008, Using natural language processing to classify suicide notes, *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, Association for Computational Linguistics, pp. 96–97.

Peter, J., Valkenburg, P. M. and Schouten, A. P.: 2005, Developing a model of adolescent friendship formation on the internet, *Cyberpsychology & behavior* **8**(5), 423–30.

Phillips, D. P. and Carstensen, L. L.: 1986, Clustering of teenage suicides after television news stories about suicide, *The New England Journal of Medicine* .

Quinlan, J.: 1993, *C4.5: Programs for machine learning*, Morgan Kaufmann, San Mateo, CA.

Raghunathan, K. and Krawczyk, S.: 2009, CS224N: Investigating SMS Text Normalization using Statistical Machine Translation, *Technical report*, Stanford University: Department of Computer Science.

Razavi, A. H., Inkpen, D., Uritsky, S. and Matwin, S.: 2010, Offensive Language Detection Using Multi-level Classification, *Advances in Artificial Intelligence*, Springer Berlin Heidelberg, pp. 16–27.

Rehurek, R. and Sojka, P.: 2010, Software Framework for Topic Modelling with Large Corpora, *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, Valletta, Malta, pp. 45–50.

Reynders, A., Scheerder, G., Molenberghs, G. and Van Audenhove, C.: 2011, Suïcide in Vlaanderen en Nederland: Een verklaring vanuit sociaal cognitieve factoren en hulpzoekend gedrag, *Technical report*, Centrum voor Zorgonderzoek & Consultancy, Leuven, Belgium.

Reynolds, K., Kontostathis, A. and Edwards, L.: 2011, Using Machine Learning to Detect Cyberbullying, *Proceedings of the 10th International Conference on Machine Learning and Applications and Workshops (ICMLA)*, IEEE.

Riloff, E.: 1995, Little Words Can Make a Big Difference for Text Classification, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, pp. 130–136.

Ritter, A., Clark, S., Mausam and Etzioni, O.: 2011, Named entity recognition in tweets: An experimental study., *Proceedings of Empirical Methods for Natural Language Processing EMNLP*, pp. 1524–1534.

Robertson, L., Skegg, K., Poore, M., Williams, S. and Taylor, B.: 2012, An adolescent suicide cluster and the possible role of electronic communication technology., *Crisis* **33**(4), 239–245.

Russell, S. J. and Norvig, P.: 1995, *Artificial Intelligence. A Modern Approach*, Prentice Hall, Englewood Cliffs, NJ.

Salton, G. and Buckley, C.: 1988, Term-weighting approaches in automatic text retrieval, *Information Processing & Management*, Vol. 24(5), pp. 513–523.

Schapire, R. E. and Singer, Y.: 2000, BoosTexter: A Boosting-based System for Text Categorization, *Machine Learning* **39**(2-3), 135–168.

Schlippe, T., Nguyen, T. and Vogel, S.: 2008, Diacritization as a machine translation problem and as a sequence labeling problem, *MT at work: Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas (AMTA-2008)*, Waikiki, Hawai'i, pp. 270–278.

Scott, S. and Matwin, S.: 1999, Feature Engineering for Text Classification, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, Morgan Kaufmann Publishers, pp. 379–388.

Sebastiani, F.: 2002, Machine Learning in Automated Text Categorization, *ACM computing surveys (CSUR)* **34**(1), 1–47.

Shapero, J. J.: 2011, *The Language of Suicide Notes*, PhD thesis.

Shneidman, E. S. and Farberow, N. L.: 1957, *Clues to Suicide*, Vol. 71, McGraw-Hill, New York.

Silverman, M. M., Berman, A. L., Sanddal, N. D., O'Carroll, P. W. and Joiner, T. E.: 2007a, Rebuilding the Tower of Babel : A Revised Nomenclature for the Study of Suicide and Suicidal Behaviors, Part 1: Background, Rationale and Methodology, *Suicide and Life-Threatening Behavior* **37**(3), 248–263.

Silverman, M. M., Berman, A. L., Sanddal, N. D., O'Carroll, P. W. and Joiner, T. E.: 2007b, Rebuilding the Tower of Babel : A Revised Nomenclature for the Study of Suicide and Suicidal Behaviors, Part 2: Suicide-Related Ideations, Communications, and Behaviors, *Suicide and Life-Threatening Behavior* **37**(3), 264–277.

Sproat, R., Black, A. W., Chen, S., Kumar, S., Ostendorf, M. and Richards, C.: 2001, Normalization of non-standard words, *Computer, Speech and Language* **15**(3), 287–333.

Stenetorp, P., Pyysalo, S., Topic, G., Ohta, T., Ananiadou, S. and Tsujii, J.: 2012, BRAT: a web-based tool for NLP-assisted text annotation, *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Avignon, France, pp. 102–107.

Stirman, S. W. and Pennebaker, J. W.: 2001, Word Use in the Poetry of Suicidal and Nonsuicidal Poets, *Psychosomatic Medicine* **63**(4), 517–522.

Stolcke, A.: 2002, Srilm - an extensible language modeling toolkit, *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP 2002*, pp. 901–904.

Tang, B., Wang, X., Wang, X., Yuan, B. and Fan, S.: 2010, A Cascade Method for Detecting Hedges and their Scope in Natural Language Text, *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, number July, Association for Computational Linguistics, Uppsala, Sweden, pp. 13–17.

Tiedemann, J.: 2012, Character-based pivot translation for under-resourced languages and domains, *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Avignon, France, pp. 141–151.

Tran, T., Luo, W., Phung, D., Harvey, R., Berk, M., Kennedy, R. L. and Venkatesh, S.: 2014, Risk stratification using data from electronic medical records better predicts suicide risks than clinician assessments., *BMC Psychiatry* **14**(76).

Treurniet, M., Clercq, O. D., van den Heuvel, H. and Oostdijk, N.: 2012, Collection of a Corpus of Dutch SMS, *Proceedings of the 8th Language Resources and Evaluation Conference (LREC'12)*, Istanbul, Turkey, pp. 2268–2273.

Van de Kauter, M., Coorman, G., Lefever, E., Desmet, B., Macken, L. and Hoste, V.: 2013, LeTs Preprocess : The multilingual LT3 linguistic preprocessing toolkit, *Computational Linguistics in the Netherlands Journal* **3**, 103–120.

van den Bosch, A., Busser, B., Daelemans, W. and Canisius, S.: 2007, An efficient memory-based morphosyntactic tagger and parser for Dutch, *Computational Linguistics in the Netherlands 2006*, Selected Papers of the 17th Computational Linguistics in the Netherlands Meeting, Leuven, Belgium, pp. 191–206.

van Heeringen, C.: 2007, *Handboek Suïcidaal Gedrag*, De Tijdstroom, Utrecht.

Van Rijsbergen, C. J.: 1979, *Information Retrieval*, Butterworth, London.

Van Royen, K., Poels, K., Daelemans, W. and Vandebosch, H.: 2014, Automatic monitoring of cyberbullying on social networking sites: From technological feasibility to desirability, *Telematics and Informatics* .

Vandekerckhove, R. and Nobels, J.: 2010, Code eclecticism: Linguistic variation and code alternation in the chat language of Flemish teenagers, *Journal of Sociolinguistics* **14**(5), 657–677.

Vapnik, V. N.: 1995, *The Nature of Statistical Learning Theory*, Springer.

Värnik, P.: 2012, Suicide in the World, *International Journal of Environmental Research and Public Health* **9**(3), 760–71.

Vilar, D., Peter, J.-T. and Ney, H.: 2007, Can we translate letters?, *Proceedings of the Second Workshop on Statistical Machine Translation*, Prague, Czech Republic, pp. 33–39.

Viola, P. and Jones, M.: 2001, Rapid object detection using a boosted cascade of simple features, *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, Vol. 1, IEEE Comput. Soc.

Vossen, P., Maks, I., Segers, R. and Van der Vliet, H.: 2008, Integrating lexical units, synsets and ontology in the Cornetto Database, *Proceedings of the 6th international Conference on Language Resources and Evaluation (LREC'08)*.

Weiss, S. M. and Kulikowski, C. A.: 1991, *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*, Morgan Kaufmann, San Mateo, California.

Whitley, D.: 1994, A genetic algorithm tutorial, *Statistics and Computing* **4**, 65–85.

Wolpert, D. and Macready, W.: 1995, No Free Lunch Theorems for Search, *Technical Report SFI-TR-95-02-010*, Santa Fe Institute, Santa Fe, NM.

World Health Organization: 2011, Causes of Death 2008 Summary Tables, *Technical report*, Health Statistics and Informatics Department, Geneva, Switzerland.

Xiang, G., Fan, B., Wang, L., Hong, J. I. and Rose, C. P.: 2012, Detecting Offensive Tweets via Topical Feature Discovery over a Large Scale Twitter Corpus, *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM)*, ACM, pp. 1980–1984.

Xue, Z., Yin, D. and Davison, B. D.: 2011, Normalizing microtext, *Proceedings of the AAAI Workshop on Analyzing Microtext*, pp. 74–79.

Yang, Y. and Joachims, T.: 2008, Text categorization, *Scholarpedia* **3**(5), 4242.

Yang, Y. and Liu, X.: 1999, A Re-examination of Text Categorization Methods, *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM Press, pp. 42–49.

Yang, Y. and Pedersen, J. O.: 1997, A comparative study on feature selection in text categorization, *Proceedings of the 14th International Conference on Machine Learning (ICML)*, pp. 412–420.

Yin, D., Xue, Z., Hong, L., Davison, B. D., Kontostathis, A. and Edwards, L.: 2009, Detection of Harassment on Web 2 . 0, *Proceedings of the Content Analysis in the WEB 2.0 (CAW2.0) Workshop at WWW2009*, Madrid, Spain.

Yu, H., Han, J. and Chang, K. C.: 2004, PEBL: Web page classification without negative examples, *IEEE Transactions on Knowledge and Data Engineering* **16**(1), 70–81.

Zhang, T. and Oles, F. J.: 2001, Text Categorization Based on Regularized Linear Classification Methods, *Information Retrieval* **4**(1), 5–31.