Phonetic Normalisation of Casual English

Cyril Mean Danilevski

51013187

c.danilevski.10@aberdeen.ac.uk

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Bachelor of Science
of the
University of Aberdeen.

Department of Computing Science

2015

Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Signed:

Date: 2015

Abstract

In this paper, a novel approach to casual text normalisation on twitter using voice technologies is proposed, based on the premise that text is partially deformed in a phonetic fashion. The concept is analysed, developed, and tested on two datasets from past publications.
We then evaluate the effectiveness of voice normalisation against these publications and found that it yields improvement when applied to existing techniques and can provide enhanced text when used on its own.

Acknowledgements

I would like to thanks Dr. Chenghua Lin for his exellent support and motivation throughout this project, Leni for being so charming (and all these cookies), and all of them homies for always being there.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

The linguist Roy Harris (b. 24 Februay 1931), once said:

> *Language is no longer regarded as peripheral to our grasp of the world we live in, but as central to it. Words are not mere vocal labels or communicational adjuncts superimposed upon an already given order of things. They are collective products of social interaction, essential instruments through which human beings constitute and articulate their world. This typically twentieth-century view of language has profoundly influenced developments throughout the whole range of human sciences.*

The extensive use of the Internet, electronic communication, and user-oriented media such as social networks, blogs or microblogs has led to the need to better understand *casually* written languages, a form of language that seldom abide to grammatical, spelling, or punctuation rules. Therefore, the *normalisation* of the language is required. Such processing poses difficulties as traditional Natural Language Processing tools and approaches are maladapted to the task (Han and Baldwin, 2011). Indeed, whether being rule based, of statistical nature, or other techniques, current approaches mostly rely on well structured text.

This problem is especially true for Twitter, the microblogging website. Despite it bustling arguably valuable data, it is underused as the language used is of poor structure (Clark et al., 2010) (Sproat et al., 2001).

This project aims to propose a novel approach for Casual English normalisation of Twitter data using speech recognition technologies.

1.2 Motivation

Normalisation, in Natural Language Processing, is the process of canonicalisation of text from one form to another. In the case of this project, the normalisation is applied from the English Twitter sociolect to formal English.

A sociolect is a register of a language associated with a social group, let it be a caste, age group, or other, regardless of the geographical backgrounds. Sociolects also differ from jargon in the sense that jargon only applies to terms and their meanings in a specific field, such as the computer,

engineering, or bobsleigh jargons. The following lists presents examples of casual text issued from Twitter:

a Aye.oops,dat was spose to be a txt
b Rndm fct bout wife: n the past 10 yrs I can cnt on one hand the num Xs she's 4gotn to unlock my car door
c OMG I LOVE YOU GUYS. You pwn:) !!!
d i need to qo to bedd qotta wakee up at 7am for school....
e heard it again! xD 3 TIMES.a sng i nvr hear!

Contrary to conventional text domains that use properly structured, or even standardised language such as (STEMG, 2013) or (Attempto Project, 2015), Twitter content is often informal and with a lesser emphasis on style. Whether intentionally crafted or not, noisy text, often created by the means of chats, emails, or, in our case, Twitter, significantly differs from the standard form of text. The desire to shorten message length as to increase typing speed but still maintain clarity characterises the structure of this text. Noisy text can be summarised as being quite the opposite of news articles that are professionally composed and edited. On Twitter, the sociolect mostly derives from spoken language. This poses a challenge for Natural Language Processing tools that are, more often than not, developed for formal languages. Therefore, if the retention of sociolects is not significant, meaning that there would be no relevant information lost, then normalisation can improve performance on downstream tasks.

Current normalisation approaches tend towards the use of either statistical machine translation, dictionary techniques, or a combined use of both.

The approach proposed in this paper is to combine the latest speech to text, speech recognition, and dictionary techniques as to ease text normalisation and improve on current techniques. As previously mentioned, the issues addressed are mostly of phonetic origin. This is based on the idea that people will not misspell text as is commonly done, but will intentionally misspell their writings in a phonetic fashion. The contribution of this project is to propose a novel normalisation approach that requires little maintenance using widely available text to speech and speech to text tools, and without requiring annotated data.

1.3 Primary Goals

Primarily, the aim of this dissertation is to evaluate the feasibility and results of using a phonetic approach to normalisation. If it turns out to be of a practical value, then a tool shall be made that may be used for further research, or simply for Twitter pre-processing. This will be considered achieved when the following will be done:

- Define the evaluation metrics for the tools to be used;
- Evaluate the performance of various Text to Speech and Speech Recognition service providers;
- Define what the purpose of *Casual English* sociolect is, for this project;
- Obtain or create a dataset of Twitter tweets;

- Obtain or create a dictionary of commonly used abbreviations;
- Perform experiments on the dataset to assess the effectiveness of the method while making necessary changes to the parameter settings for a better optimisation and performance.
- Compare these results with the results of relevant publications.

1.4 Secondary Goals

A secondary goal, if time permits, is to not only study the English language, but to also try and evaluate this technique on other languages. For this, French was selected.

In French, the language is similarly marred, offering a sociolect that is comparable to English. It would be therefore interesting to see what results the software yields in this language. To do this, the following will need to be done:

- Obtain or create a French tweet dataset;
- Evaluate the software's outcome as is done for the English language;
- Compare the results of both languages.

1.5 Structure

This document is organised as follows. This chapter introduces the topic, its motivations and goals.
Chapter 2, Background and Related Work, presents the problematic, the existing techniques used, and works related to the problem.
Chapter 3, Requirements, specifies both functional and non-functional requirements.
Chapter 4, Problem Domain, describes in more depth and more technically what had to be done prior to beginning the implementation.
Chapter 5, Methodology and Technologies defines how the project was conducted, and what technologies were used.
Chapter 6, System Design and Architecture, explains how is the software built.
Chapter 7, Testing, describes the testing made to ensure the reliability of the software.
Chapter 8, Evaluation, reports and evaluates the results from the software.
Finally, the report finishes with chapter 9, Summary & Conclusions, which are then followed by the bibliography and appendices.

Chapter 2

Background and Related Work

2.1 Existing Techniques and Related Work

A survey from the literature revealed that generally, three approaches are commonly used: machine translation, dictionaries, or both simultaneously.

2.1.1 Statistical Machine Translation

One way to undertake Casual English normalisation is to consider it not as a sociolect, but as a language of its own, as did (Ling et al., 2013).

Statistical Machine Translation is often used between two closely related languages (German and English for instance), it may therefore be intuitive to make use of it for the normalisation of a sociolect. These are based on linguistic information of both the source and target languages. Given an input sentence, such a system will output a sentence in the target language based on morphological, semantic, and syntactic analyses of both languages.

The most commonly used tool is Giza++ (Och and Ney, 2003), freely available online under the GPLv2 license. This tool permits the training of statistical translation models and provides various alignment models.

2.1.2 Dictionaries

Text normalisation is often considered similar to spell checking (Han and Baldwin, 2011), hence the extensive use of this methodology in many papers throughout the years: (Rowe and Laitinen, 1995), (Choudhury et al., 2007), (Jadhav et al., 2013).

Whilst most of the mentioned papers use the idea itself and develop their own software, only (Clark and Araki, 2011) have used actual spell checkers with custom dictionaries. Using the open source softwares Aspell and Hunspell, they showed that the average error per sentences in their experiments decreased from fifteen percents to less than five percent.

2.1.3 Combined Methods

The aggregation of both aforementioned techniques has revealed to remove the majority of noise from a tweet, and increase its readability significantly. In their paper, (Kaufmann and Kalita, 2010) made use of dictionaries to pre-process the most commonly misspelled words, doing orthographic normalisation, and then further processed the tweets with the Giza++ tool. The author reports an increase of eighteen percents in his BLEU scores after normalisation with this technique.

2.2 Speech Technologies

Speech Technologies are nowadays widely in use, mostly as assistive technology to enable disabled people to make use of information technologies. This project aims at taking advantage of the advances in this field.

2.2.1 Speech Synthesis

The analysis for the correct pronunciation of written text is one of the most challenging tasks in speech synthesis. With the recent expansions in the field, much work has been done to resolve the issues it presents. This is the constituent this project aims to explore.

Text normalisation, in its tradition sense, is the first step of a speech synthesis system, where numbers, dates, and acronyms found in real-world text are converted into actual words, such that the system may then pronounce these correctly. Spell checking plays an important role in this process. Many suggested to use the noisy channel framework, first proposed by Shannon (Shannon, 1948) to generate a list of corrections for misspelled word, ranked by the corresponding posterior probabilities. (Sproat et al., 2001) enhanced this framework by calculating the likelihood probability as the chance of a noisy token and its associated tag being generated by a specific word.

2.2.2 Speech Recognition

The use of speech processing for normalisation has previously been investigated and assessed by (Kobus et al., 2008), then rapidly discarded as it was considered inadequate and, since then, the idea has been abandoned. However, I strongly believe that this is due to the fact that, back in 2008, speech technologies were still in their infancy and relatively unexplored. Nowadays, with the advanced work and development on Google Now and Apple Siri, I suggest to review this approach.

A somewhat related idea has been explored in (Beckley and Roark, 2013), where the authors have created a "pronunciation checker": similar to a spell checker, this dictionary approach generates possible phonemes sets for a written token, and selects the most likely occurrence. In the paper, this technique has yield interesting and valuable results, but requires extensive work to implement.

2.3 The Casual English Sociolect

This section aims to define what is meant by Casual English, and which of the issues that it presents are pertinent and addressed in the scope of this project.

In their paper, (Clark and Araki, 2011) define eight characteristics of Casual English. The motivation for such categorisation is that these are distinct problems that may be resolved independently or are covering different semantics. This classification is of use in this project as it permitted a clear identification of the project's relevant issues. These characteristics are described as follows:

> *Our Casual English Conversion System (CECS) is designed on the basis that errors and irregular language used in casual English found in social media can be grouped into several distinct categories, and accordingly, a multi-faceted approach will be the most effective way to deal with the problem. The categories used in CECS' database are as follows.*

1. Abbreviation (shortform). *Examples: nite ("night"), sayin ("saying"); may include letter/number mixes such as gr8 ("great").*
2. Abbreviation (acronym). *Examples: lol ("laugh out loud"), iirc ("if I remember correctly"), etc.*
3. Typing error/misspelling. *Examples: wouls ("would"), rediculous ("ridiculous").*
4. Punctuation omission/error. *Examples: im ("I'm"), dont ("don't").*
5. Non-dictionary slang. *This category includes word sense disambiguation (WSD) problems caused by slang uses of standard words, e.g. that was well mint ("that was very good"). It also includes specific cultural reference or in group-memes.*
6. Wordplay. *Includes phonetic spelling and intentional misspelling for verbal effect, e.g. that was soooooo great ("that was so great").*
7. Censor avoidance. *Using numbers or punctuation to disguise vulgarities, e.g. sh1t, f***, etc.*
8. Emoticons. *While often recognized by a human reader, emoticons are not usually understood in NLP tasks such as Machine Translation and Information Retrieval. Examples: :) (smiling face), <3 (heart).*

With these definitions, the approach aims at resolving 1, short form abbreviations, 2, acronyms, 4, punctuation omissions or errors, and 6, wordplays.

Three of these categories, 1, 4, and 6, in my opinion, stem from the same origin. The spreading of electronic communication, from the 1990s onwards, has brought about the need for fast and, from a human point of view, more practical forms of written text.
Mostly composed from mobile devices (originally being SMS) with each their specific constraints, the aim of such aberrations is to reduce the length (to match the maximum characters allowance), and to enhance the input (from impractical devices) of written language.

It may be amusing to the reader to note that this tendency is not only limited to social medias, but is also widely used in the professional world of programming, as shows the matlab function *num2str()*, read out as *"num to string"*, or the python function *str()* read out as *"string"*, which returns a string representation of the given argument.

The remaining category 2 can be addressed using a dictionary.

2.4 Comparison with Past Publications

In order to assess the effectiveness of this approach, it will be benchmarked against two publications regarding the normalisation of Twitter messages in English. Both of the selected publications provide the tweet datasets used for their experimentations, and the matching golden standards.

In their paper, (Han and Baldwin, 2011) use a combination of dictionary lookups, word similarity, and context support to achieve a nearly perfect BLEU score (0.934). However, they do not attempt to normalise casual English into formal English and neither do they attempt to correct misspelt words. Instead, they concentrate their effort on remediating out-of-vocabulary (OOV) tokens, that is, words that are not recognised as a part of the formal English vocabulary, but are parts of the previously defined sociolect.

Therefore, the golden standard provided with the dataset is not a reference of formal English sentence, but contains the tweets with *only* the OOV words annotated. For instance *wit* will not be normalised to *with*, as it is a misspelling that may be addressed with an off the self spell checker, but *u* will be normalised to *you* or *ttyl* to *talk to you later*. Thus, it may prove difficult to match their exactness and high scoring, as it addresses only the point 1 and 2 (short form abbreviations and acronyms) of the sociolect as defined earlier on.

The other publication that this project is compared to is (Pennell and Liu, 2014). In this paper, the authors make use of statistical machine translation techniques to achieve normalisation of tweets. Contrary to the previous paper, these authors attempt normalisation toward formal English, and this is reflected in the quality of their golden standard.

With both of these publications, we cover the results two of the major types of normalisation methodologies at the current state of the art.

Chapter 3

Requirements

The requirements described here are only a fraction of the metric used for evaluating the completeness of the project.
As mentioned in chapter 1, the aim of this project is to evaluate the use of speech technologies as a tool for natural language preprocessing. Therefore, as the following functional requirement define what the finished software is, the chapter 8, Evaluation and Testing, will define whether the technique is refined enough once the outcome of evaluations will be satisfactory.

3.1 Functional Requirements

Table 3.1: The System's Functional Requirements

No.	Description	Priority
1	**Analysis**	
1.1	Initialise connection with speech services provider	High
1.2	Initialise a number of threads	Low
1.3	Split the data into a number of queues	Low
2	**User Input**	
2.1	File input support	High
2.1.1	Read a single tweet per line.	High
2.2	Web interface input support	Medium
3	**Output**	
3.1	File output support	High
3.1.1	Write a single tweet per line in the same order as input file	High
3.2	Web interface output support	Medium

Due to the amount of data to be processed, it is vital for the system to be scalable. The bottleneck resides not in resource intensive work, such as CPU access where there often are race conditions, or networking latencies, but merely in the time taken to resolve a normalisation. Whilst normalising none to, let us say, fifty entries, it may be insignificant to do multiprocessing, and

easier to carry the task one entry at a time. But on tasks where there are many more entries, sequentially processing them becomes a limiting factor. Therefore, threads are used.
The quantity of threads to be used often is a major and difficult question to answer. Testing showed that having more threads than CPU cores available slows the process as time is spent switching between threads. Therefore, the quantity of threads used is determined by the quantity of cores available on the device running the software.

The main purpose of the web interface is to be used for demonstration and rapid experimentation. Web browsers are ubiquitous, and web application lightweight. It also means that no modifications have to been made to the software, as it is used as is from the web application.

3.2 Non-Functional Requirements

As this system is dependent on APIs provided by Google, it was crucially important to devote a considerable amount of time to testing, to ensure that this step did not affect the overall performance of the system and the quality of its results. The system's scalability is another important aspect as it would normalise several hundreds of tweets. The process would therefore be more time consuming as well as require more computational resources. It is important that the system handles the data load well.

Since the project relies on internet connectivity to solve normalisation, an internet connection of better reliability than the university's eduroam is required.

Chapter 4

Problem Domain

4.1 Speech Tools Selection

There are several providers of speech synthesis and recognition available. The tools reviewed in this project are described in table 4.1.

Table 4.1: Overview of the considered services providers.

		Description	Notes	Source
Google	TTS SR	Available from their translation services	Web based API	Google (2015)
Apple	TTS SR	Available from Siri	Requires an Apple device running Mac Os X	Apple (2015)
iSpeech	TTS SR	Proprietary TTS and SR service provider, but requires a fee to use	Web based API	iSpeech (2015)
AT&T	TTS SR	Free access to TTS and SR available, but heavily limited	Web based API	AT&T (2015)
Microsoft SAPI	TTS SR	Available from MSDN	Requires a device running MS Windows	Microsoft (2015)
Festival CMU Sphinx	TTS SR	Open-source software From Carnegie Mellon University	Free, multi platform	CSTR (1999) Lamere et al. (2003)

All of the aforementioned text to speech tools use the same methodology.
All decode each syllable separately and then concatenate the resulting utterances into words. For example, *"B4"* is processed as the two syllables /ˈbiː/ (bee), /ˈ fɔər/ (four) resulting in the string *"Before"*, whereas *"GR8"* is processed as /ˈ dʒiː/, /ɑr/ (arr), /ˈ eɪt/ (eight), creating the string *"Gee Arr Eight"* as the three characters are not strictly representing syllables. Therefore, some pre-processing will be required. Single digits that are parts of the token will be converted to their literal forms. As such, the token *gr8* will become *greight*, and will be captured as a single syllable word, thus circumventing the issue.

The choice of speech synthesis tool was made in accordance with the best speech recognition tool as most providers offer both.

4.1.1 Evaluation

A test was conducted with 500 out of vocabulary tokens, from the (Han and Baldwin, 2011) set, that were judged by a panel of three students to be easily recognisable as belonging to category 1 (short form abbreviations) and 6 (wordplays) as described in chapter 2.
The three students were all female, in their mid-twenties, knowledgeable with and users of social networks, one native speaker from England, and two European foreigners of estimated (by the author) of level C1/C2 on the CERF Scale (Verhelst et al., 2009). They were rewarded with a trip to *Foodstory Cafe*, a coffee shop in Aberdeen.

The system were evaluated on their ability to correct the out of vocabulary tokens.

After initial testing, of which the results are shown in table 4.2, Siri and Google Translate, which uses the technological back-end of Google Now, showed to be the best performing tools. This can probably be justified as these are widely used by thousands of users worldwide, thus providing extensive training to the systems, resulting in these being highly robust to noisy text and speech.

Accurary Result	
Google	68%
Siri	65%
AT&T	43%
Microsoft SAPI	56%
Festival / CMU Sphinx	48%

Table 4.2: Evaluation results over 500 out of vocabulary tokens.

Subsequent testing with more tweets and further out of vocabulary tokens belonging to other categories revealed that Google Translate outperforms Apple Siri with a thirty percent improvement on yielding the correct normalised words. It seems, albeit no data has been collected, that Siri is better at understanding natural voices, whilst Google manages better synthesised voices. It could be that, if the tweets were to be read aloud by a person, then Siri would outperform Google.

With these results, it was decided to use Google's technologies for the development of this project.

4.2 Data Acquisition

4.2.1 English Tweet Dataset

As mentioned in chapter 2, two datasets from previous publications are used, both of which provide a matching golden standard.

4.2.2 English Normalisation Dictionary

The normalisation dictionary used for this project was compiled from the (Jones, 2015) database. This database was created and is maintained by its author since 2005. As the data is freely available on the website in HTML format, a parser was written to convert the information in a more usable comma separated values file format. This database was used as not only is it exhaustive, but is also used in the aforementioned papers. It contains 3607 entries.

Since the slangs and abbreviations used on the internet are constantly evolving, keeping a static dictionary seems problematic. Therefore, as well as using the compiled dictionary, an access to abbreviations.com's API (Abbreviations.com, 2015) has been obtained. This website offers an up to date dictionary that is crowd-sourced, but moderated by company staff.

4.2.3 French SMS Dataset

Since a readily available corpus of tweets in French could not be found, it was first decided to compile a new one. However, the task of creating a golden standard would still have to be carried through. Since it highly time consuming, it was decided to instead use a SMS set. Such messages

are highly similar to tweets in form and structure. A corpus of eighty eight thousand donated text messages is available from (Panckhurst et al., 2014). Only a fraction of these, thirty-two, were used in the scope of this project. This subset was created by the author as to ensure that most of the data contains misspelling and out of vocabulary tokens, as well as respect the length limitation of text messages. Several text messages can be sent together as one in order to overcome the 160 characters limit.

Ten of these messages were chosen as they fully match the annotated reference. Five were chosen as they do not fully match the annotated reference, but have slight deviations (missing accents, involuntary misspellings), the remaining contain out of vocabulary tokens.

Note that these messages contain only French word. It is indeed very common in this language to proceed to what is called *anglicism*, the use of English word as substitution for their French equivalent. For instance the word *maquillage* is often replaced with its English equivalent *make-up*, or *experiment* replacing *expérience*.

4.2.4 French Normalisation Dictionary

The normalisation dictionary used for this language was compiled from various websites across the internet, as no one source on its own was complete. It was desirable to create a dictionary comparable to the English one in size, but only 1013 unique entries could be found.

4.3 Speech Processing of Tokens

Early in the design stage was it necessary to define how to use the speech tools for normalisation. The original hypothesis was that submitting a complete tweet for speech processing would be the optimal solutions as it would provide context for the tokens to be normalised, thus enhancing the probabilistic model created by the tools. To verify this hypothesis, it was decided to submit whole tweets, tri-grams, and unigrams.

Tri-grams were used as they would include the two words immediately surrounding the token to be normalised. If the token were to be at the beginning or end of a tweet, then the two preceding or following words were used for the formation of the tri-gram. The test was conducted using the same 500 out of vocabulary tokens as mentioned in the above section 4.1.1, Speech Tool Selection.

	Complete tweets	Tri-grams	Unigrams
Accuracy score	13.5%	43.78%	63.56%

Table 4.3: Results of testing of various lengths of spoken text for out of vocabulary tokens.

From table 4.3, it can be seen that, despite the original assumption, it is better to submit only a single token. The system was therefore designed with a unigram approach.

Chapter 5

Methodology and Technologies

This chapter describes the methodology used in this project for the research, design, implementation and testing. It also tells about the technologies used to achieve these goals.

5.1 Methodology

The plan aimed to finish the project by the 23rd of April 2015, thus having one week spare in case of issues arising. However, even though the work was generally following the original guideline, some stages took longer than was initially planned as their thorough completion was found to be particularly important for further progression. Figure 5.1 shows a revised timeline of the main project activities.

- Literature Review: 15/01/15 - 04/02/15
- Implementation: 26/01/15 - 06/03/15
- Testing: 02/02/15 - 20/03/15
- Report Writing: 30/01/15 - 23/04/15

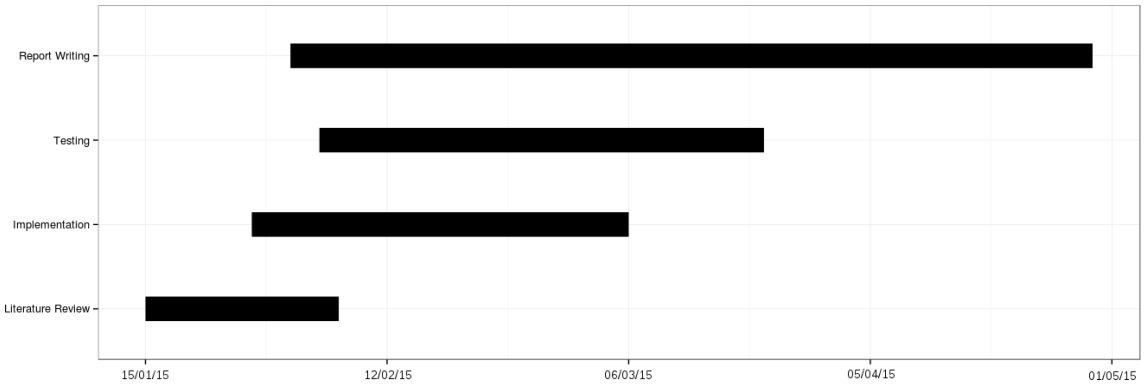

Figure 5.1: Project Timeline

The AGILE approach was employed during the implementation phase of the project, as the requirements evolved with each iteration, based on the weekly meetings with the supervisor and the outcome of previous testing.

To ensure a smooth transition between the different hardware used, as well as prevent the losses in case of corruption, regular backups were made using the git version control system on servers in various locations.

Documentation was considered first and foremost, with the belief that good documentation helps in the writing of software. Thus, once an idea reached maturity on paper, the documentation was written before the implementation, and then review to ensure that the software does indeed what is intended and that the documentation reflects it.

5.2 Technologies

The use of development technologies was determined by the desire to keep the software platform independent.

5.2.1 Python

Although originally prototyped in C, the project was written in the Python 2.7 programming language for a number of beneficial reasons. First, Python provides operating system independent bindings ensuring that the software could be used seamlessly on any host supporting the language without the need for modification or intricate conditional operations.

Additionally, this allows the software to be used as a library for further usage without the need for any modification.

5.2.2 Development Tools

The work was developed on a computer with an Intel i5-3320M running at 2.6 Ghz, and 4 Gb Ram.

Vim, bpython, pep8, and git were the tools used for development. The choice to use a GNU/Linux system for the development was motivated by the ease of programming on these platforms. This platform has historically been designed with development in mind, and the most comfortable tools blend nicely with the working environment that POSIX provides.

5.2.3 Libraries and Pre-Requisites

As to identify the words that are out of vocabulary, a spell checker is required. PyEnchant, (Kelly, 2015), is a spell checking library for Python. It combines all the functionalities of the underlying Enchant library with the flexibility of Python and an object-oriented interface. Another reason for its use is the fact that it was used in the reference paper introduced earlier on.

In order to convert audio files of the generated speech, FFMPEG, (Bellard et al., 2015), is used. This tool was chosen as it is simple, convenient, and readily available on numerous platforms.

There exist no official API to access Google speech generation nor recognition. However, it operates in a RESTful fashion and examples of its use can be found on the internet. Google Translate at its 38th stage (launched December 2014) is used for this project.

The web interface is powered by the python web framework flask version 0.10.1 (Ronacher, 2015) with the addition of jQuery (Resig, 2015) for the formatting of user inputs. These two softwares are commonly used for developing lightweight web applications. Furthermore, little to none overhead work is required with these frameworks in order to make one's application web based.

In order to evaluate the system's results, the software iBLEU, version 2.6.2, is used. This software is described in (Madnani, 2011). This software is used as it provides not only BLEU scoring for NLP tasks, but also insightful visualisation of the scoring, thus enabling developers to

pinpoint where are the software's weaknesses.

Whilst most resources are available for all platforms, one of the reviewed technology is Apple's Siri. In order to use this technology, an Apple Computer is required. My supervisor, Dr. Chenghua Lin, has obtained from Dot.Rural a machine for the duration of the project.

5.3 Risk Assessment

It is possible that the selected speech services provider becomes unavailable, due to licensing restriction for instance. In that case, I shall revert to using open-source tools, such as the previously mentioned festival, even though these may be of lesser accuracy or practicality. Other software used as libraries and for development are all under some form of GPL license, thus reducing limitations for third party usage and ensuring that the software, in the version used, will remain open of access.

Another similar risk is that, with further updates from the service providers, the quality of normalisation will vary, perhaps for the better, perhaps with inferior results. This is because these tools, whilst doing some light normalisation, are not designed for the specific task of normalisation.

Finally, there exist a risk that the project will not see to completion, as the task at hand may prove too difficult. The solution to such an issue, however, is to work harder and reevaluate the requirements with the project's supervisor.

Chapter 6

System Design and Architecture

This chapter details the system's design and architecture.

Given the variety of problems to solve, as described in section 2.3: The Casual English Sociolect, it was decided to solve the different issues individually with a divide and conquer methodology. In order to unify the processes, the software is organised in two parts: the dictionary part, and the speech part.

Firstly, the dictionary will attempt to resolve the most common misspelling, as these are more easily recognisable, and the out of vocabulary words that are not recognised by the dictionary will then be resolved using the aforementioned speech tools. Figure 6.1 illustrates the normalisation of text.

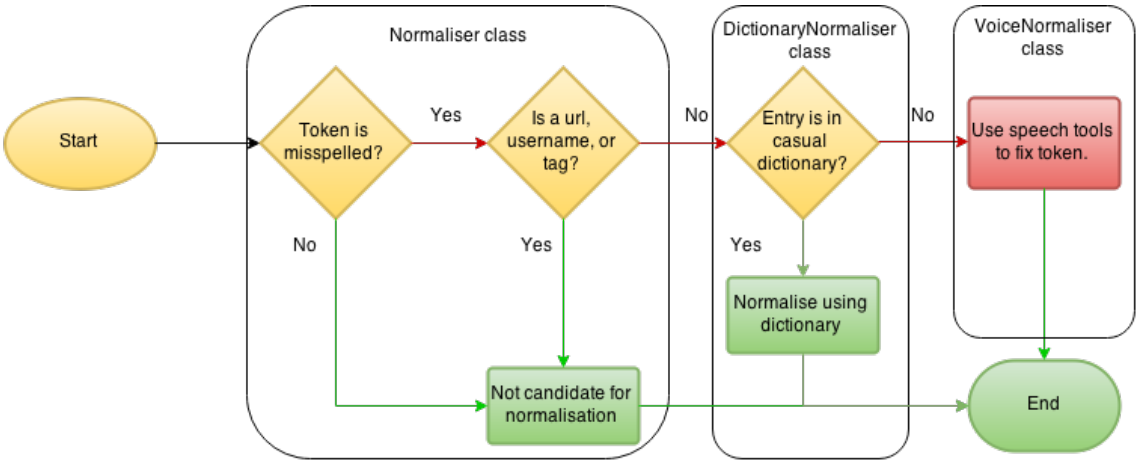

Figure 6.1: Flowchart of the processing of text.

The decision to operate in this way was motivated by experimentation. The various flow considered were:

1. Speech normalisation applied to the outcome of dictionary normalisation;
2. Dictionary normalisation applied to the outcome of speech normalisation;
3. Apply both normalisation and choose the most appropriate outcome;
4. Attempt speech normalisation, and if the result is deemed inappropriate, attempt a dictionary normalisation;
5. Attempt dictionary normalisation, and if the result is not conclusive, attempt a speech normalisation.

Applying one type of normalisation to the outcome of another, as suggested in point 1 and 2, yield results that often were very far off the correct form.
Attempting speech normalisation first could be of use, but defining whether the spoken form is the proper correction is a hard problem with natural languages. Moreover, if the correct form is already known, it is better to reduce the use of statistical guessing.
Likewise, using both forms of normalisation at the same time, and then choosing the most correct output is a difficult task. These could be solved using techniques such as calculating the Levenstein distance between the outcomes and the original misspelt token, or by comparing the phonetic difference, with algorithms such as Soundex, double metaphone or NYSIIS. However, trials showed this approaches to be of little success.

Thus, the remaining work flow solution of first proceeding to a dictionary and, if to no avail, use speech normalisation.

6.1 Architecture

As each of the components described in figure 6.1 are to be individually actionable, it is sensible to split the software in distinct parts that do each a single task. Each component (whether or not to use dictionaries, which dictionary to use, should the speech normalisation be used, what language is the normalisation done with...) can be activated or deactivated by the use of booleans that may be set either during run time, or can be set in a static fashion by setting the flags at launch.

6.1.1 Initialisation

This is to be considered the main class of the system. It will split a tweet into tokens (groups of characters separated by white spaces), identify misspelt tokens, and verify whether theses are twitter specific (usernames or tags). These tasks are a part of the *normaliser.py* file as they are independent of either normalisation techniques employed.

6.1.2 Dictionary Normalisation

Dictionary Normalisation starts once a word is judged misspelt. Two dictionaries can be used. The first, compiled by the author, is stored in a csv file with two columns, the first representing the known slang term and the second the English term. This file is then read into a python dictionary object. The second dictionary is provided by (Abbreviations.com, 2015) in the form of a RESTful API access.
This API allows the specification of the family of abbreviations to lookup. These can be communication, computing science, or law, for instance. However, this is not used in this project, but instead the most popular abbreviation is obtained.
Either dictionary can be enabled or disabled. However, when both are enabled, the definition from the local file will be used if it exists, otherwise the remote dictionary is queried.

6.1.3 Voice Normalisation

If the dictionary normaliser does not provide a solution, then the voice normaliser will attempt to provide a correct normalised token.

The text to speech generates a flac audio file, but speech recognition makes use of mp3. Therefore the file has to be converted. This is achieved using FFMPEG, (Bellard et al., 2015).

6.1.4 Results

The results are presented in three different ways. If a file of tweets were given as an argument, then the output would be saved in a file of the same name with the *n_* prefix. Since the input file must contain a single tweet per line, the normalised output will be found on the same line number as a given input. If the input is given as a string from the standard input (or command line), then the result will be written to the standard output. Finally, if the web interface is used, then the output of normalisation will be displayed on the web page.

Chapter 7

Testing

This chapter reports the testing strategies used both during and after the development stage of the project.

7.1 Components

Testing the various components was an ongoing activity, and progressed and refined as the software was developed. Agile and black-box testing were used. Each component of the program was tested with typical and erroneous inputs and their outputs were verified for correctness. This testing was reiterated until all errors were discovered and resolved and the component behaved as expected for all typical inputs.

7.2 Integration

Integration testing was considerably simplified by the fact that the program was divided into well-defined parts, as described in section 6.1 and figure 6.1. After every part of the program was developed and unit tested, it was integrated with the rest of the system. Following the methodology of component testing, the updated system was verified anew for issues that may have arisen. Due to the system's increasing complexity, more valid and invalid inputs were given, to ensure that the stability of the whole did not fail. If some crucial parts were unimplemented at the time of integration, these were replaced with stubs that simulated the desired behaviour in a simplified manner. As the development progressed the stubs were progressively replaced with finished components.

7.3 Stability and Performance

As the program is expected to run for prolonged periods of time, it was necessary to perform stability testing as the overall software approached finalisation. The program ran for prolonged periods of time whilst logging debugging information. Several errors, such as the mishandling of unicode characters were found.

For instance, the software was originally developed using the ASCII standard, with the English language in mind. However, it was found that the datasets used were encoded in ISO-8859-1, also known as Latin-1. This standard is can be considered a superset of ASCII as it supports many more latin characters (191 in total). This led to reoccurring errors when writing the normalised text to files.

This early catch allowed to speed up and ease the further processing of French later on.

7.4 Speed and Efficiency

While the speed and efficiency of the system were not a primary concern, it was important that they be considered during the implementation. Since the bottleneck of the software resides in the time taken to resolve a tweet, and not on input or output operations, it was decided early on to make use of multi-threading. Also, since the system makes use of internet connectivity, it is understandable that the system's performance will vary over time. After many iterations of testing and modifications, the system is now running at a satisfactorily pace.

Chapter 8

Evaluation

With the previous stages of testing completed, the software was free from technical errors, but its suitability for the task at hand has yet to be assessed. This section will introduce the evaluation metrics used and the results of the tests conducted.

Three sets of evaluations were conducted, each with different datasets. The first uses the dataset provided by (Han and Baldwin, 2011), then the evaluations are repeated using the dataset from (Pennell and Liu, 2014). Finally, the tests are conducted using the French set of SMS described in section 4.2.3.

8.1 Evaluation Metrics

In order to measure the effectiveness of this system, an evaluation metric needs to be defined. In this case, two different metrics will be used. First, the Word Error Rate (WER) is used, as it provides direct and intuitive results on the system's performances.
Then, using the BLEU scoring method, it is possible to compare this system with the paper the previously referred to (Han and Baldwin, 2011).

Word Error Rate is typically used to measure the performances of speech recognition systems. However, it has also been used for text normalisation tasks. It is derived from the Levenshtein distance measurement, but instead of working at the level of characters, it evaluates words compared to a reference text, also known as golden standard. WER indicates the rate of incorrectly normalised words, the lower the rate (at a minimum of zero percent), the better the normalisation. This rate is not limited and can be over a hundred percent, if there are many word insertions for instance, in the case of lower quality normalisation.

The Bilingual Evaluation Understudy algorithm, BLEU, is mostly used to evaluate machine translation tasks from one language to another. It aims to measure the quality of text by calculating the score of individually translated sentences, or tweets in this experiment, against the golden standard, and then averaged over the whole corpus to provide an estimate of the overall normalisation quality. BLEU Score is quick to use, inexpensive to operate, language independent, and correlates highly with human evaluation. It is the most widely used automated method of determining the quality of machine translation.

The reason behind using WER, as well as BLEU, is that not only does it provide intuitive results, but also that the golden standard provided with (Han and Baldwin, 2011) cannot be considered good English and does not compare to a translation into a properly constructed English. The golden standard was made by fixing individual words in tweets, but not tweets as a whole.

Thus, the grammar and the word ordering is still somewhat dubious. If this report were written in such language with such grammar, the markers would score it rather badly indeed. Moreover, there is no guarantee of a higher score indicating an improved normalisation quality.

The software used to calculate the WER was developed for this project by the author.

iBLEU, the software used to calculate the BLEU score was obtained from (Madnani, 2011). This software is presented in the form of a web interface running onto one's localhost and provides instant BLEU scoring given the original input file, the normalised output file, and the golden standard. It then offers a visual representation of the scoring of each sentences such that one may examine one's results in details to explore where are one's approaches' pitfall.

8.2 Exploratory Data Analysis

To get a clearer understanding of the data before normalisation, as well as help the understanding of the results, it is important to understand the range and quality of the data used. Exploratory Data Analysis, EDA, consists of analysing and summarising data. The data that are relevant for this project are: the quantity of messages in a set, the average length of a message, the quantity of tokens in the set, the quantity of out of vocabulary token, and the quantity of correct word.

	Set Size	Average Message Length (in words)	Tokens in Set	Out-of-Vocabulary Tokens	Correct Tokens
Han and Baldwin set	549	18.28	10576	1184*	9392*
Pennel and Liu set	4661	4.69	21876	7769	14107
Panckhurst et al. subset	30	10.86	326	88	238

Table 8.1: EDA of the three datasets used, showing their structures.
* denotes as reported by the authors, and are not targeting words recognised by a spell checker as English, but misspelt.

Table 8.1 reports the analysis of each set. It shows that the Han and Baldwin set is constituted of longer tweets and has the highest OOV to token ration between the English sets. The Pennell and Liu set is larger, but contains shorter messages on average, which may lead to higher quality messages, as having less to write in a message may reduce the desire to abbreviate or reduce text input.

The French set is smaller, for reasons explained in section 4.2.3, but contains nonetheless a sizeable amount of out of vocabulary and misspelt tokens that are representative of mistakes commonly done in this language.

8.3 Experimental Setup

In the evaluation of the system on each of the sets, the following tests are conducted: a baseline test, a speech normalisation test, a dictionary only normalisation, and the combination of both techniques.

The baseline test consists of obtaining both the BLEU score and the WER from the original set before any normalisation or modification against the golden standard. This provides a reference score of the worst case scenario, showing what to improve on and how is the data originally deformed. Moreover, this will provide measurable difference when normalisation techniques will be applied.

Then comes the voice normalisation evaluation. In this test, all the tokens that are considered misspelt are attempted to be normalised using the speech tool. This will provide use with another

valuable measurement, as this project consists mainly of finding out whether speech normalisation is a viable methodology.

The dictionary normalisation alone is then tested, yielding yet another measurement which, in itself, is not that important to the project, but allows us to measure the effect of dictionary normalisation in a combined approach.

Finally, the combined use of the two techniques is evaluated, and the results are then summarised and analysed for the dataset.

8.4 English Tweet Normalisation

8.4.1 The Han and Baldwin Set

Baseline Test

In order to effectively judge the effect of the processes on the data, it is required to establish a baseline measurement showing how is the unmodified data different from the target golden standard. Figure 8.1 represents the BLEU score for each individual sentence and table 8.2 summarises these results. From these results, an overall score of 0.237, we can observe that the original tweets are hardly conforming to traditional English grammar and spelling, compared to the golden standard.

Type	1-gram	2-gram	3-gram	4-gram
Individual	0.4674	0.3523	0.2550	0.1761
Cumulative	0.4238	0.3672	0.3154	0.2374

Table 8.2: Results of BLEU scoring without any mean of normalisation on the Han and Baldwin set.

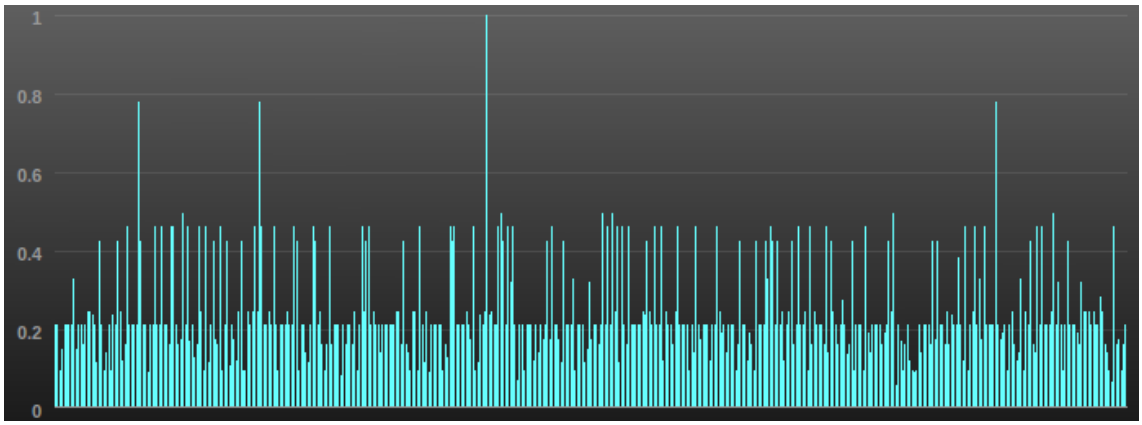

Figure 8.1: Plot of baseline test results on the Han and Baldwin set.

Voice Normalisation

As the aim is to enhance normalisation using a phonetic approach, the measurement of speech normalisation needs evaluation. Results are reported in figure 8.2 and table 8.3.

These already show an improvement upon the baseline test. However, whilst the results seem, intuitively, much improved, as shown in figure 8.2, there is still room for improvement. After a more in depth evaluation of the results, it appears that fixing abbreviations from the internet chat sociolect, one-to-multiple relations (*sooooo* actually being *so*) are often failing, resulting in

other substitutions being inserted. However, it is good at normalising issues such as punctuation omissions (*didnt* to *didn't*) and commonly used abbreviations, some of which would be harder to do using a dictionary (*st.* meaning either *saint*, *street*, or *such that*).

Type	1-gram	2-gram	3-gram	4-gram
Individual	0.8071	0.6749	0.5642	0.4743
Cumulative	0.7916	0.7239	0.6619	0.6060

Table 8.3: Results of BLEU scoring using speech normalisation on the Han and Baldwin set.

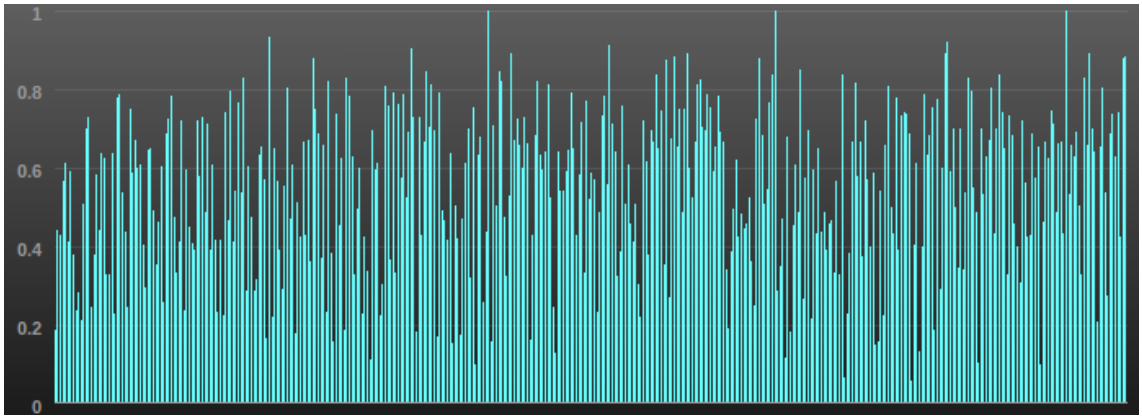

Figure 8.2: Plot of speech normalisation results on the Han and Baldwin set.

Dictionary Normalisation

Now that the score for speech normalisation on its own has been established, it is required to establish the effect of dictionary normalisation on its own. The results of this evaluation are reported in table 8.4 and in figure 8.3. From these results, it was observed that obscure abbreviations, related to online chats, as well as certain one-to-many relations are successfully normalised. Yet the entries in the dictionary do not take grammar into account, but address only spelling and abbreviations. Lastly, it will normalise only the tokens available in the dictionary, thus requiring an ever larger dictionary, resulting in a database that is hard to manage.

Type	1-gram	2-gram	3-gram	4-gram
Individual	0.9120	0.7728	0.6574	0.5637
Cumulative	0.7436	0.6845	0.6309	0.5829

Table 8.4: Results of BLEU scoring using dictionary normalisation on the Han and Baldwin set.

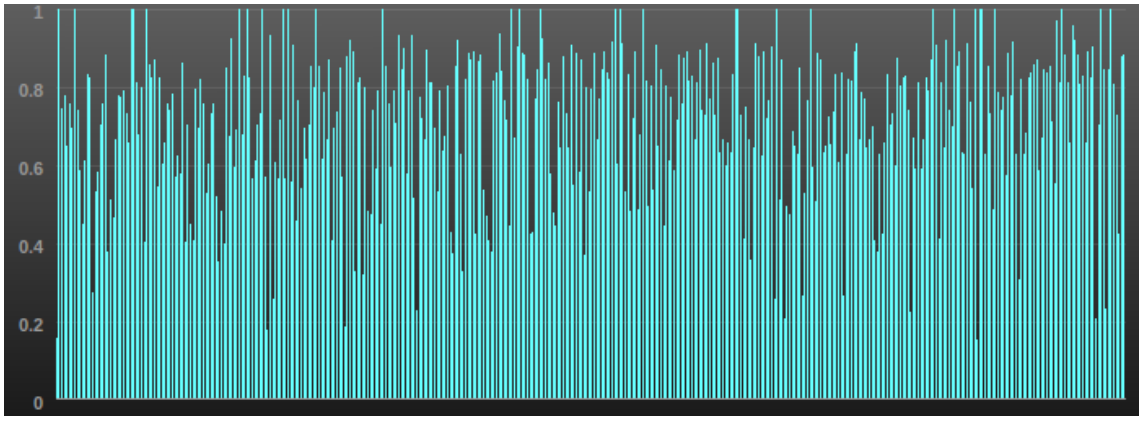

Figure 8.3: Plot of dictionary normalisation results on the Han and Baldwin set.

Combined Normalisation

The results of the combination of both methodologies, as depicted in figure 6.1, are reported in table 8.5 and figure 8.4. These results, when analysed more closely, reveals that the resulting grammar is inconsistent and words are sometimes substituted by other non-related words, due to the speech recognition tool. Nonetheless, both commonly used abbreviations and more obscure ones are being successfully addressed, as well as many of the punctuation errors.

Type	1-gram	2-gram	3-gram	4-gram
Individual	0.8970	0.8066	0.7267	0.6577
Cumulative	0.8891	0.8432	0.8001	0.7601

Table 8.5: Results of BLEU scoring for normalisation using both techniques on the Han and Baldwin set.

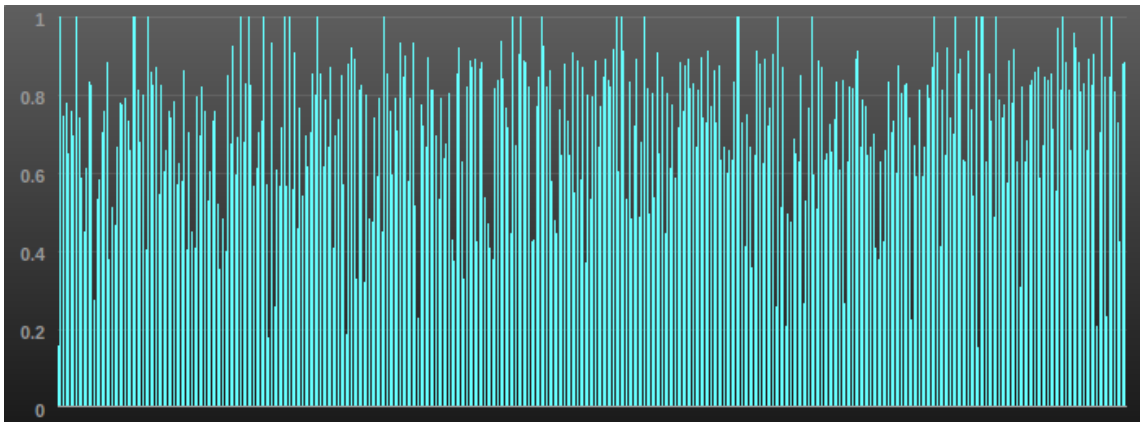

Figure 8.4: Plot of the combined normalisation procedures results on the Han and Baldwin set.

Overview

The overall results are reported in table 8.6. The BLEU score of the reference paper is also reported.

The system provides an increased BLEU score compared to the non-normalised text. Whilst not as accurate as the approach proposed in (Han and Baldwin, 2011), we still manage to increase

	WER	Insertions	Deletions	Substitutions	BLEU
Original					0.237
Dictionary Normalisation	25.56	1653	13	439	0.582
Speech Normalisation	26.81	443	321	1444	0.606
Combined Normalisation	13.89	218	94	828	0.760
(Han and Baldwin, 2011)					0.934

Table 8.6: Overall results, including WER, compared to the (Han and Baldwin, 2011).

the quality of text by an important factor. It is observed that the addition of dictionary normalisation to the process increases the overall quality of textual normalisation.
On the downside, most one-to-many relations are failed to normalise.

8.4.2 The Pennell and Liu Set

In this section, the tests are repeated using the Pennell and Liu dataset. This will yield interesting results, as the golden standard, is of a higher standard, carefully prepared by five annotators, from the Tweeter sociolect to formal English. An important difference, previously highlighted, is that this golden standard is literally *translated* into formal English, including grammar, punctuation, spelling, the clauses carried by the tweets, and the syntax.

Therefore, obtaining a high BLEU score will prove harder than with the previous set.

Baseline Test

The baseline test, reported in table 8.7 and figure 8.5, shows that the tweets are all quite similar one to another, yet still far from the golden standard with a BLEU score of 0.075.

Type	1-gram	2-gram	3-gram	4-gram
Individual	0.7792	0.4857	0.0232	0.0037
Cumulative	0.7792	0.6152	0.2064	0.0753

Table 8.7: Baseline BLEU without normalisation applied.

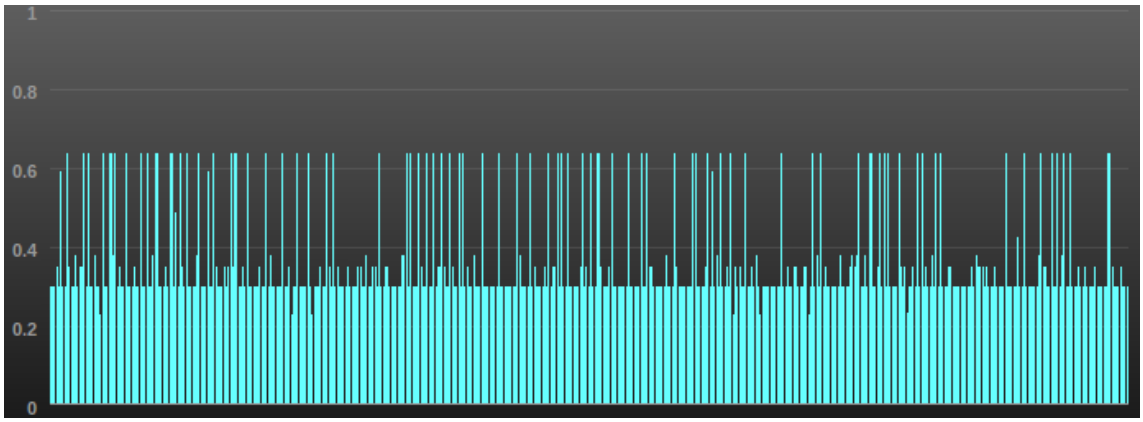

Figure 8.5: Plot of baseline test results.

Speech Normalisation

Attempting speech normalisation shows a noticeable improvement in regards to the golden standard, bringing several tweets close to a hundred percent correlation with the golden standard. The BLEU score is increased from 0.075 to 0.264, almost quadrupled from the baseline test. The processing showed a general improvement of the grammar that was not expected, given the results yield by the previous dataset. Obscure abbreviations coming from the twitter sociolect were not normalised properly, as well as certain words being replaced by irrelevant words, such as *Annie* substituted for *Any*.

Type	1-gram	2-gram	3-gram	4-gram
Individual	0.7306	0.4667	0.1275	0.1130
Cumulative	0.7306	0.5839	0.3516	0.2648

Table 8.8: Results of BLEU scoring using voice normalisation on the Pennell and Liu set.

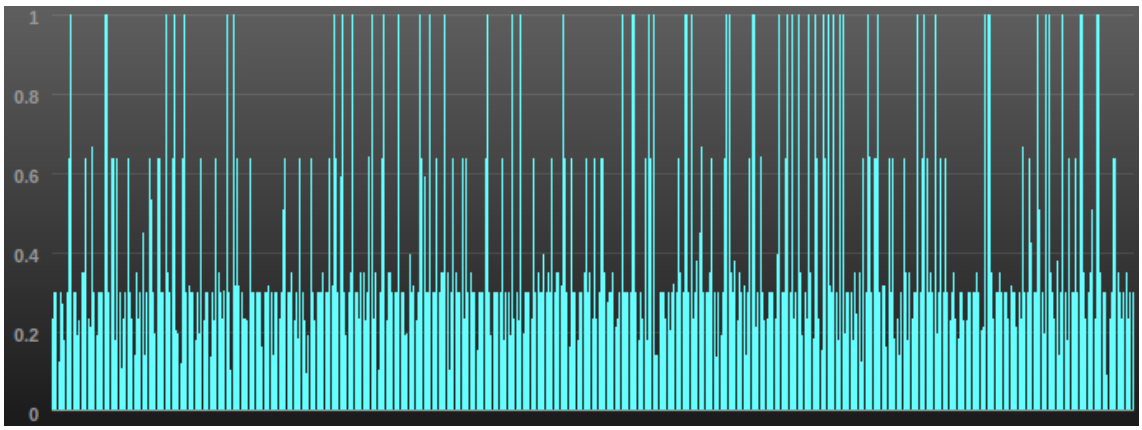

Figure 8.6: Plot of speech test results.

Dictionary Normalisation

Dictionary normalisation improved little the corpus, scoring only 7.51 with BLEU, being even lesser than the baseline measurement of 7.53. This is probably due to expressions found in the dictionary that were not normalised in the golden standard or perhaps replacing wrongly valid tokens.

Type	1-gram	2-gram	3-gram	4-gram
Individual	0.7769	0.4833	0.0232	0.0036
Cumulative	0.7769	0.6127	0.2057	0.0751

Table 8.9: Results of BLEU scoring using dictionary normalisation on the Pennell and Liu set.

Combined Normalisation

Using the combined form of normalisation confirms the results obtained from the previous two test, that dictionary normalisation, with the dictionary that was previously compiled, has little effect on this set but speech normalisation increases the quality of text by a noticeable margin,

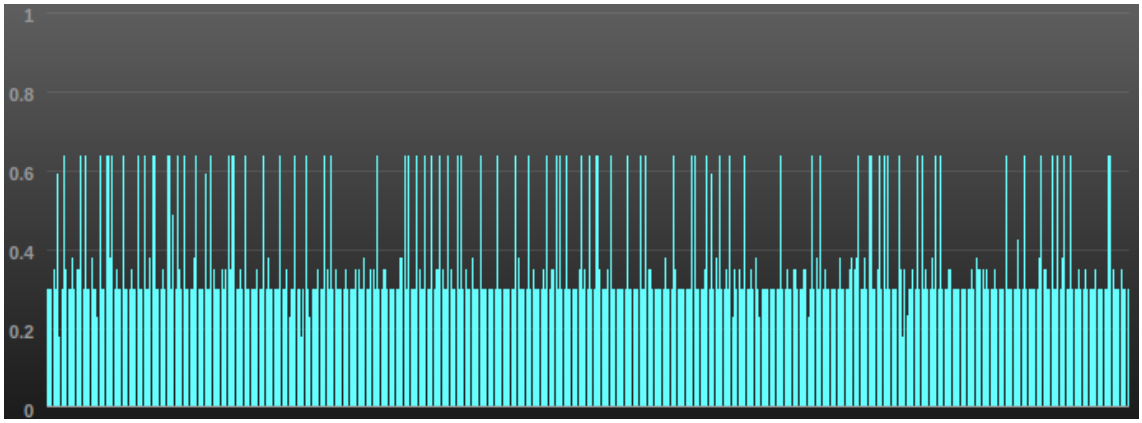

Figure 8.7: Plot of dictionary test results.

going from a BLEU score of 0.075 to an improved 0.264. This is the same result as in the previous test of speech normalisation alone.

Type	1-gram	2-gram	3-gram	4-gram
Individual	0.7306	0.4667	0.1275	0.1130
Cumulative	0.7306	0.5839	0.3516	0.2648

Table 8.10: Results of BLEU scoring for normalisation using both techniques.

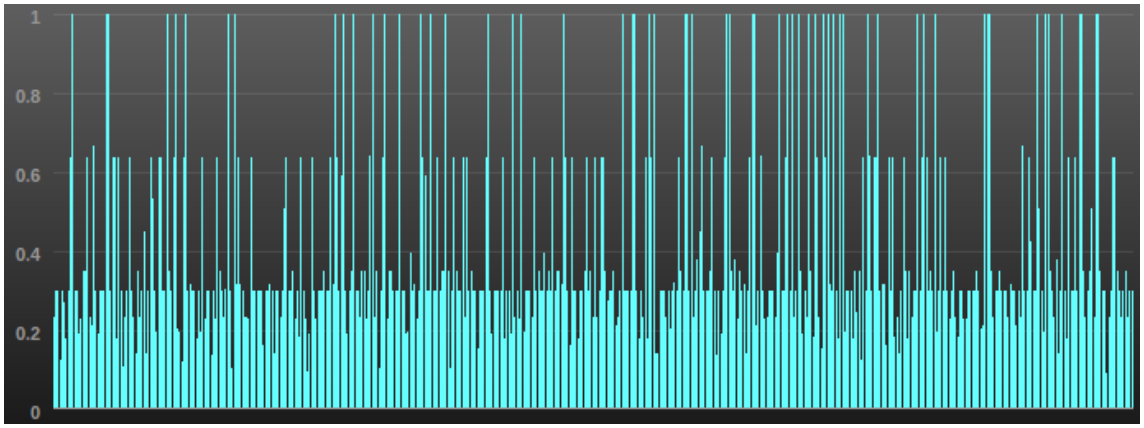

Figure 8.8: Plot of the combined normalisation procedures results.

Overview

The overall results are reported in table 8.11. The BLEU score of the reference paper is also reported. Since the dictionary did little changes, voice normalisation provided an extensive step

	WER	Insertions	Deletions	Substitutions	BLEU
Original					0.075
Dictionary Normalisation	28.31	0	87	568	0.075
Speech Normalisation	22.77	0	9	518	0.264
Combined Normalisation	22.77	0	9	518	0.264
(Pennell and Liu, 2014)	6.8				

Table 8.11: Overall results, including WER, compared to the (Pennell and Liu, 2014).

forward the golden standard. The principal issues with the normalised output are similar as these described in voice normalisation alone.

This is still a much lower score than the results presented by the authors, but it is though to be higher than would the (Han and Baldwin, 2011) methodology be applied to this set.

8.5 French SMS Normalisation

For linguistic curiosity, the methodology is applied to the French set of SMSs. This measurement is more for an informal evaluation on how well this methodology carries on other languages. French has its language modified in a similar phonetic fashion in short messages.

Baseline

The baseline measurement of this set seems more promising than its English counterparts. However, this is quite a small set and may not be representative of the superset it comes from. The BLEU score for the baseline test is 0.447.

Type	1-gram	2-gram	3-gram	4-gram
Individual	0.7893	0.6573	0.5447	0.4473
Cumulative	0.5919	0.5401	0.4921	0.4471

Table 8.12: Results of baseline BLEU scoring for the French set.

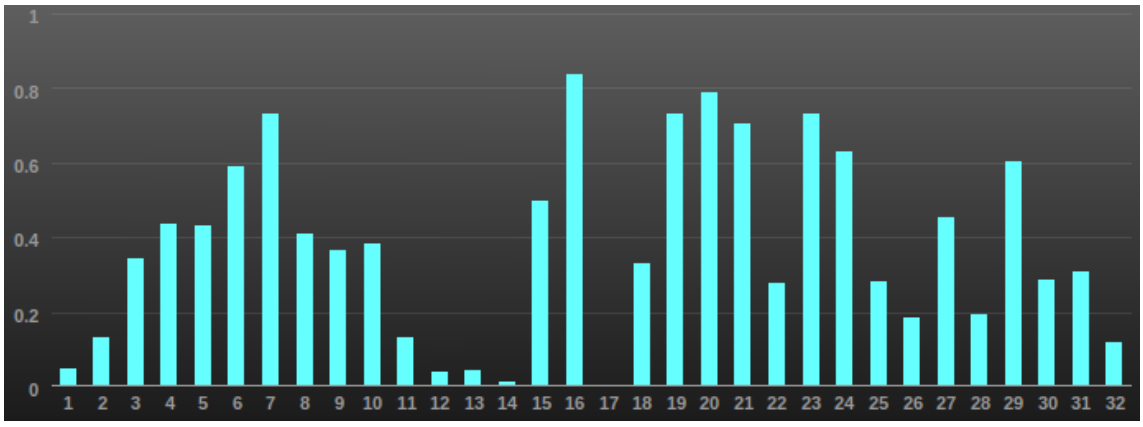

Figure 8.9: Plot of the baseline results in French.

Voice Normalisation

Voice normalisation made results that were already higher, with a score of 0.531. Contrary to the results obtained in English, voice normalisation contributed to an increased correctness in grammar. Many of the common grammatical mistakes were fixed, such as *ses*, *ces*, *c'est* that are similar to the *your* and *you're* mistake in English. The incorrect substitution of words seen in English was not present. It is, as well, performing better than dictionary normalisation, as reported below.

Type	1-gram	2-gram	3-gram	4-gram
Individual	0.7992	0.6685	0.5606	0.4651
Cumulative	0.6954	0.6360	0.5822	0.5316

Table 8.13: Speech normalisation results on the French language.

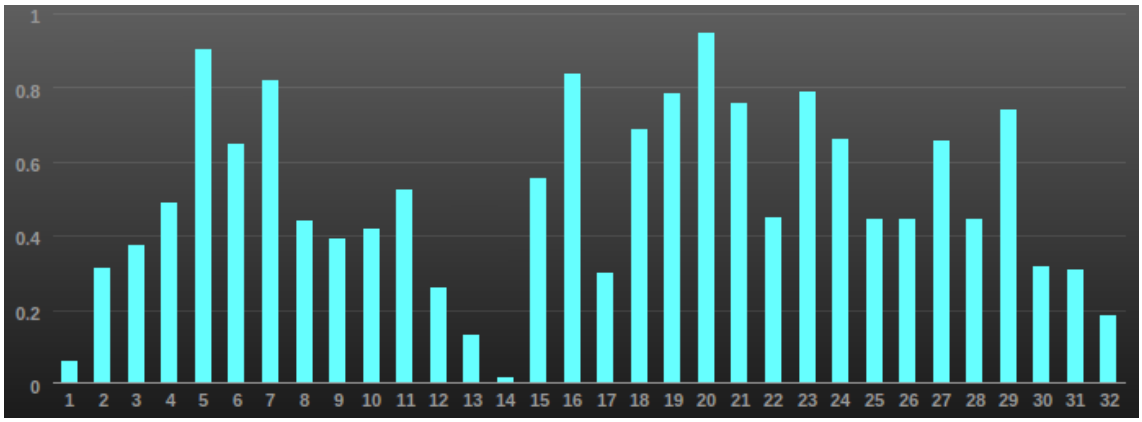

Figure 8.10: Plot of the speech normalisation results in French.

Dictionary Normalisation

Dictionary normalisation, in table 8.14 and figure 8.11, did improve the corpus, albeit not as much voice normalisation. This is another example where dictionary normalisation may fail: with many spelling mistakes, but little abbreviations, dictionary normalisation cannot catch all mistakes unless a comprehensive, but difficult to create and maintain, dictionary is used, as only the entries present in the dictionary can be addressed.

Type	1-gram	2-gram	3-gram	4-gram
Individual	0.7923	0.6602	0.5521	0.4553
Cumulative	0.6266	0.5719	0.5227	0.4762

Table 8.14: Dictionary normalisation results on the French set.

Combined Normalisation

Combined normalisation worked as expected, dictionary and voice processing working along to increase the quality of text. Although it is hard to be definitive, as the set is too small to be fully representative of the language, it appears that this technique applies as well in French as in English.

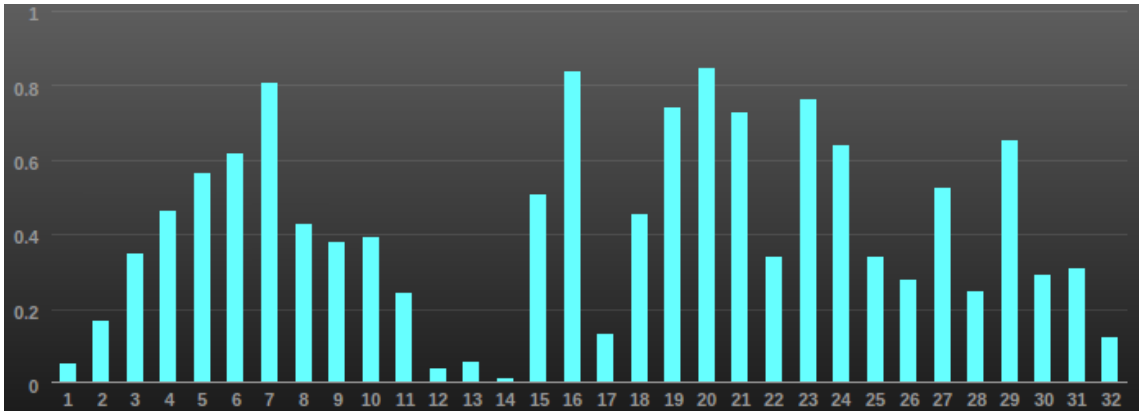

Figure 8.11: Plot of dictionary normalisation results in French.

Type	1-gram	2-gram	3-gram	4-gram
Individual	0.8000	0.6658	0.5601	0.4680
Cumulative	0.7298	0.6658	0.6096	0.5576

Table 8.15: Results of BLEU scoring using the combined normalisation procedures in French.

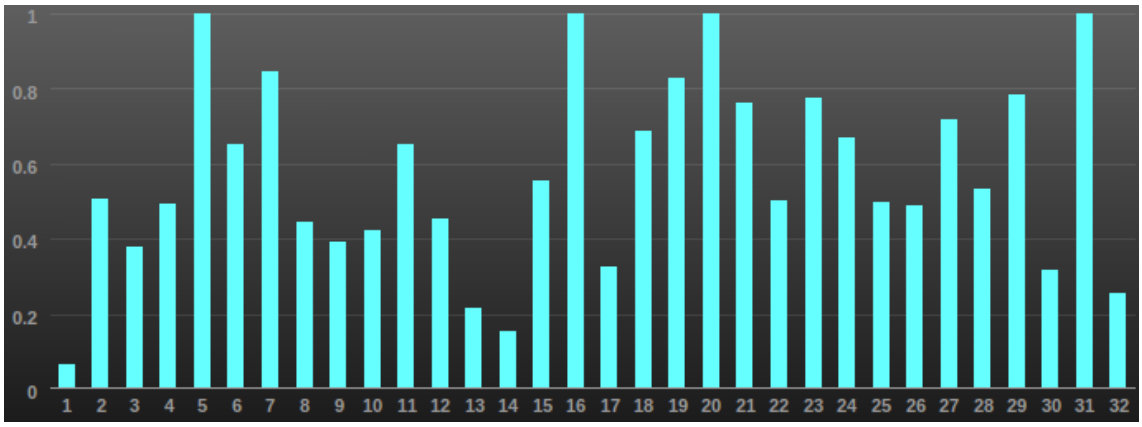

Figure 8.12: Plot of the combined normalisation procedures results in French.

Overview

The application of this methodology was similarly tried and tested on the French SMS set. However, there hasn't been time to find literature to compare the results in that language with.

The overall results are slightly less satisfactory than of English normalisation. This seemingly stems from the quality of the dictionary. Indeed, it has been harder to compile dictionary as valid and extensive as the English dictionary, resulting in a lesser accurate French dictionary. A second justification, however unlikely, is that because the French speaking population is lower than the English speaking, and with a lesser range of accents and expressions, it is harder for the text to speech and speech recognition systems to manipulate ill-formed words and text. This explanation is unlikely as in French, voice normalisation worked slightly better as it did in English. This, I think, is because the French language has less phonemes and other "legal" sounds, and therefore more grapheme combinations can lead to the same sound, leading to less pronunciation possibilities.

	WER	Insertions	Deletions	Substitutions	BLEU
Original					0.447
Dictionary Normalisation	43.30	191	3	158	0.476
Speech Normalisation	37.15	132	3	167	0.531
Combined Normalisation	34.32	100	3	176	0.557

Table 8.16: Overall results, including WER, for the normalisation applied to the French SMS set.

8.6 Results Review

With the complete data from the two English sets, it becomes apparent that it is difficult, using this approach, to normalise the text towards formal English, as reported in the Pennell and Liu findings, whereas normalising the text in a *"good enough"* way, where a perfect text isn't required but merely need to be clear enough to understand the meaning conveyed, such as with the Han and Baldwin set, this normalisation procedure proves to be of practical use.

Interestingly, voice normalisation showed a great increase in the quality of text with the Pennell and Liu set, whereas it did little improvements on the Han and Baldwin set. Meanwhile, dictionary normalisation yield opposite results. From these, we can conclude that phonetic normalisation has its place amongst other more widely used normalisation techniques. Like all of the techniques mentioned in chapter 2: Background, voice normalisation needs to be used where fit and a careful choice of the tool to be used needs to be done.

Table 8.17 summarises these findings.

From all the points we wished to address, only one remains to address. Words with one-to-multiple relations, as defined in point 6, wordplays, in section 2.3, appeared to be harder to master, due to their lack of consistency. A potential algorithm that could address the issue is to remove duplicate letters until a legal combination is obtained.

Thus, from the word *gooood*, we could obtain the closest valid word *good*. However, another valid candidate is *god*. This specific issue could be addressed using context support, or perhaps hidden markov models. However, at the time of writing this report, the University of Groningen, under the supervision of Rob van der Goot, is offering a two year research masters to resolve this single issue of text normalisation.

With this knowledge, it is more understandable that the issue proved unresolvable within the time frame of this project.

		Advantages	Drawbacks
Han and Baldwin set	Voice Normalisation	Common abbreviations/misspellings	Obscure abbreviations/misspellings; one to multiple relations
	Dictionary Normalisation	Obscure abbreviations/misspellings, certain one to multiple relations	Does not fix grammar; only words in dictionary; requires large dictionary
	Combined Normalisation	Both of the above	Results in incoherent grammar; sometimes words a being substituted with unrelated words
Pennell and Liu set	Voice Normalisation	Better grammar	Obscure abbreviations; misunderstandings between names and words (any/Annie)
	Dictionary Normalisation	Obscure abbreviations/misspellings, certain one to multiple relations	Does not fix grammar; only words in dictionary; requires large dictionary
	Combined Normalisation	Resolves several out of vocabulary issues.	Results in incoherent grammar
Panckhurst et al. subset	Voice Normalisation	Better grammar (ces/ses/s'est)	Anglisism; many misunderstanding
	Dictionary Normalisation	Obscure abbreviations/misspellings, certain one to multiple relations	Does not fix grammar; only words in dictionary; requires large dictionary
	Combined Normalisation	Both of the above	Anglisism; many misunderstanding

Table 8.17: Performance review across the different sets

Chapter 9

Summary And Conclusion

9.1 Summary

In this project, we have defined several normalisation criteria and have attempted to resolve these issues with varying degrees of success.

Looking back at the project's goals and requirements, it is possible to say that almost all of them were successfully carried through.

- **Define the evaluation metrics for the tools to be used**

  Whilst reviewing the literature, it became apparent that two metrics are widely used, BLEU and WER, and these were thus used for comparing with publications.

- **Evaluate the performance of various Text to Speech and Speech Recognition service providers**

  This has been done, and revealed surprising results. It was originally though that using the more common Apple products would yield better results. But Google seem to have the edge.

- **Define what the purpose of Casual English sociolect is, for this project**

  Several publications defined their own understanding of what textual normalisation should achieve. The most relevant and detailed definition was found in (Clark and Araki, 2011) and subsequently used for this project.

- **Perform experiments on the dataset to assess the effectiveness of the method while making necessary changes to the parameter settings for a better optimisation and performance**

  This process was ongoing for the duration of the project in order to obtain the best possible results. For each iteration, a specific normalisation task was targeted. This led most of the points addressed to be resolved.

- **Compare these results with the results of relevant publications**

  This was done once the application was finalised. Two paper that published both their results and datasets were used. The two paper had differing sets, thus providing a greater range of data with which to benchmark the quality of the system.

We have two interesting and differing results: in one case, we found on the (Han and Baldwin, 2011) set that speech normalisation did not achieve much on its own, but worked great as a complement to dictionary normalisation. On the other hand, with the (Pennell and Liu, 2014) set, which contains less abbreviation specific to the twitter domain and where the dictionary on its own was inefficient, speech normalisation resulted in much better text.

9.2 Future Work

There are many ways to develop the project further. Certain of the idea concern the project as is, other concern the tried approach.

Keeping the generated audio files in a folder with each file named with the tweet ID is of interest as one could analyse in more depth how was the original token perceived.

Originally, it was decided to make use of threads in order to increase processing speed. However, more time than expected has been spent on the development and testing of the methodology, and it was thus deemed that there would not be enough time to test the system thoroughly with the added complexity of threading. Therefore, threading can be added.

The fundamental premise of this project's approach is that people misspell in a phonetic fashion. Therefore, instead of using speech directly, one could attempt phonetic normalisation, as suggested by the work of (Beckley and Roark, 2013). Their idea of a pronunciation checker could be developed further by converting the individual syllables into, for instance, the International Phonetic Alphabet, and then back to English. This, correlated with the use of algorithms such as Double Metaphone, might yield desirable results.

Alternatively, instead of using off the shelf speech tools, that are designed for formal English, it is possible to design a pronunciation dictionary, as is possible for the Festival system.

9.3 Conclusion

Most of the primary and secondary goals were achieved. The developed framework used the concepts described in this report and the tool tested on several datasets. The results are not ideal, but really, in the world of automation, the aim is often not to be as good as fellow humans doing the job, but merely to be good enough.

Our hope is that the rise of other technology, such as word prediction, could reduce the use of abbreviations and intentional misspelling, thus reducing the needs for sociolects, although it is not clear such technology is widely used and, moreover, it seem fundamentally human to constantly come up with new inventive ways to use tools, let it be to reach for space beyond the stars or communicating one's enthusiasm in a more accentuated way. And thus suggestions for further investigation were proposed.

Bibliography

Abbreviations.com (2015). Abbreviations.com, the web's largest resource for acronyms and abbreviations. http://www.abbreviations.com/. Online; accessed 24/02/2015.

Apple (2015). Apple siri. http://www.apple.com/ios/siri/. Online; accessed 28/02/2015.

AT&T (2015). At&t natural voices. http://www2.research.att.com/~ttsweb/tts/demo.php. Online; accessed 28/02/2015.

Attempto Project, U. o. Z. (2015). Attempto controlled english (ace). http://attempto.ifi.uzh.ch/. Online; accessed 29/01/2015.

Beckley, R. and Roark, B. (2013). Pair language models for deriving alternative pronunciations and spellings from pronunciation dictionaries. In *EMNLP*, pages 1584–1589.

Bellard, F., Niedermayer, M., et al. (2015). Ffmpeg. http://ffmpeg.org. Online; accessed 07/02/2015.

Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S., and Basu, A. (2007). Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(3-4):157–174.

Clark, E. and Araki, K. (2011). Text normalization in social media: progress, problems and applications for a pre-processing system of casual english. *Procedia-Social and Behavioral Sciences*, 27:2–11.

Clark, E., Roberts, T., and Araki, K. (2010). Towards a pre-processing system for casual english annotated with linguistic and cultural information. In *Proceedings of the Fifth IASTED International Conference*, volume 711, pages 044–84.

CSTR (1999). Centre for Speech Technology Research, University of Edinburgh. http://www.cstr.ed.ac.uk/. Online; accessed 28/02/2015.

Google (2015). Google translate. http://google.com/translate. Online; accessed 28/02/2015.

Han, B. and Baldwin, T. (2011). Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 368–378. Association for Computational Linguistics.

iSpeech (2015). ispeech. http://ispeech.org. Online; accessed 28/02/2015.

Jadhav, S. A., Somayajulu, D. V., Bhattu, S. N., Subramanyam, R., and Suresh, P. (2013). Topic dependent cross-word spelling corrections for web sentiment analysis. In *Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on*, pages 1093–1096. IEEE.

Jones, R. (2015). Slang dictionary - text slang and internet slang words: A guide to everyday

acronyms and obscure abbreviations. http://www.noslang.com/dictionary/. Online; accessed 08/02/2015.

Kaufmann, M. and Kalita, J. (2010). Syntactic normalization of twitter messages. In *International conference on natural language processing, Kharagpur, India*.

Kelly, R. (2015). Pyenchant, a spellchecking library for python, version 1.6.6. http://pythonhosted.org/pyenchant/. Online; accessed 24/02/2015.

Kobus, C., Yvon, F., and Damnati, G. (2008). Normalizing sms: are two metaphors better than one? In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 441–448. Association for Computational Linguistics.

Lamere, P., Kwok, P., Gouvea, E., et al. (2003). The cmu sphinx-4 speech recognition system. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2003), Hong Kong*, volume 1, pages 2–5.

Ling, W., Dyer, C., Black, A. W., and Trancoso, I. (2013). Paraphrasing 4 microblog normalization. In *EMNLP*, pages 73–84.

Madnani, N. (2011). ibleu: Interactively debugging and scoring statistical machine translation systems. In *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*, pages 213–214. IEEE.

Microsoft (2015). Microsoft speech api (sapi) 5.3. https://msdn.microsoft.com/en-us/library/ms723627%28v=vs.85%29.aspx. Online; accessed 28/02/2015.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Panckhurst, R., Détrie, C., Lopez, C., Moïse, C., Roche, M., and Verine, B. (2014). 88milsms. a corpus of authentic text messages in french. Produced by l'Université Paul-Valéry Montpellier 3 and CNRS, in collaboration with l'Université catholique de Louvain, financed with the support of MSH-M and the Ministère de la Culture (Délégation générale à la langue française et aux langues de France, with the participation of Praxiling, Lirmm, Lidilem, Tetis, Viseo.

Pennell, D. L. and Liu, Y. (2014). Normalization of informal text. *Computer Speech & Language*, 28(1):256 – 277.

Resig, J. (2015). jquery: write less, do more. http://jquery.com/. Online; accessed 02/03/2015.

Ronacher, A. (2015). Flask (a python microframework). http://flask.pocoo.org/. Online; accessed 02/03/2015.

Rowe, N. and Laitinen, K. (1995). Semiautomatic disabbreviation of technical text. *Information Processing & Management*, 31(6):851–857.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Sproat, R., Black, A. W., Chen, S., Kumar, S., Ostendorf, M., and Richards, C. (2001). Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

STEMG, A. (2013). The official home of asd simplified technical english, asd-ste100 (ste). http://www.asd-ste100.org/. Online; accessed 29/01/2015.

Verhelst, N., Van Avermaet, P., Takala, S., Figueras, N., and North, B. (2009). *Common European Framework of Reference for Languages: learning, teaching, assessment*. Cambridge University

Press.

Appendix A

User Manual

This appendix presents the user manual for the software.

A.1 Pre-requisites

In order to use the system, the following is required:

1. Python 2.7 or newer;
2. FFMEG 1.2.6 or newer;
3. PyEnchant 1.6.6 or newer;
4. Flask 0.10.1 or newer [1];
5. Internet connectivity.

A.2 Using the System

As the software is written in Python 2.7, there is no need for compilation, as this is an intepreted language.

A.2.1 Command Line

The system is meant to be used from a command line or any preferred standard input/output system. The software expect either the `-s` or `-f` flag. `-s` indicate that the following string is candidate for normalisation, and the output will be sent to the standard output. `-f` indicates that the following string is the name of the file with text to normalise.

```
./normaliser.py -f filename
./normaliser.py -s str
```

The input file is expected to have a single tweet per line. The software will then create a file prexifed `n_` with the normalised text.

A.2.2 Web Interface

The web interface's purpose is for demonstration only or quick testing. Once all the pre-requisites are met, you may use the web interface by executing the following command:

```
./web_norm.py
```

[1] This is only required for the web interface

Then, pointing a web browser to the specified url (`http://localhost:5000/`) will open the web interface when testing the application can be done.

Entering a string in the field will return the input normalised using the three approaches of dictionary normalisation, voice normalisation, and both combined.

A.2.3 External Library

The software can be used as a library for integration with further Python projects. The either of the three methods (voice, dictionary, or combined) can be used as follows:

```
import normaliser as n
import dictionary_normaliser as dn
import voice_normaliser as vn

sample_string = "Helo m8"

normaliser = n.Normaliser(debug=False)
dict_norm = dn.DictionaryNormaliser(debug=True)
voice_norm = vn.VoiceNormaliser(debug=True)

print normaliser.normalise(sample_string) #prints "hello mate"
print dict_norm.normalise(sample_string)  #prints "Helo mate"
print voice_norm.normalise(sample_string) #prints "Hello m8"
```

Listing A.1: Import example

A.2.4 Using a Specific Normalisation Method

Each module can be used independently, in order to use only a single normalisation method, for instance.

Dictionary normalisation alone is done by calling the `dictionary_normalisation.py` file.

Likewise, calling the `voice_normalisation.py` file will result in voice normalisation only applied to the input text.

Appendix B

Maintenance Manual

This appendix presents the maintenance manual for the software. The software is split in three separate files. These represent the different stages as shown in figure 6.1. Each module can be used independently, in order to use only a single normalisation method, for instance. The source code, together with sample data, is located in the `source` folder in the project submission.

B.1 Software Pre-requisites

In order to use the system, the following is required:

1. Python 2.7 or newer;
2. FFMEG 1.2.6 or newer;
3. PyEnchant 1.6.6 or newer;
4. Flask 0.10.1 or newer [1];
5. Internet connectivity.

B.2 Hardware Pre-requisites

Approximatively 25Mb of disk space and up to 100MB of memory are required during runtime. The amount required varies depending on the settings used.

B.3 Installation

To install the software, extract the sources from the archive.

B.4 Temporary Files

Two temporary audio files are created whilst using voice normalisation. An MP3 of the generated speech is originally created. It must then be transcoded into FLAC for the speech recognition engine to process the data. The FLAC file is deleted at the end of the processing of a token. If the boolean `thorough` is set to `True` in the `_cleanup` function, then the MP3 file will be deleted as well. Otherwise, it is overwritten at the next token normalisation.

[1] This is only required for the web interface

B.5 Source Code List

The project consists of the following source files:

File Name	Description
normaliser.py	The main file of the software, contains the `Normaliser` class
dict_normaliser.py	contains the `DictionaryNormaliser` class
voice_normaliser.py	contains the `VoiceNormaliser` class
web_norm.py	A flask and jQuery web interface to the software

Table B.1: List and description of the source files.

B.6 Known Issues

If the internet connection fails, it is known that the software will abruptly exit and not clean the temporary files. The system will have to be manually restarted from the last processed tweet.