

LS-UG-6 Fixing Twitter

Alex Burley
Lucia Specia
COM3600

May 3, 2016

This report is submitted in partial fulfilment of the requirement for the degree of Computer Science by Alex Burley.

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations which are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name:

Date:

Signature:

Abstract

Everyday hundreds of millions of people open up their Twitter application and write their thoughts to the world. From teenagers in school to famous celebrities, people of all backgrounds can interact with each other across the internet by sending short messages or simply giving indication of approval by the press of a button.

Twitter has become a central data source for areas of research like sentiment analysis and machine translation. However, Twitter has no unified language and each 'tweet' has a character limit that leads users to emphasize quality over quantity. The problem arises that tweets are full of useful information yet hidden in a mass of non-standard terms created by web culture. Emoticons, slang, abbreviations, all of these terms that are so easily understandable from human view can cause large problems for machine translation systems.

This project's objective is to examine this irregular use of English language, specifically in the context of tweets, and implement various methods to bring each tweet closer to its true representation in standard English. Using regular expressions we can categorize each component of a tweet and use these techniques to 'normalize' the component to regular language. Through this, we aim to improve the ability for a public machine translation system to translate irregular tweets into a foreign language.

Acknowledgments

I would like to thank my supervisor, Lucia Specia, for guiding me throughout this project and also my housemates for keeping me sane this past year.

Contents

1	Introduction	1
1.1	Introduction to Twitter	1
1.2	The Problem	1
1.3	Summary	2
2	Literature Review	3
2.1	Analysis of a 'Tweet'	3
2.2	Data Sources	4
2.3	Classifying Tokens	5
2.4	Normalisation Techniques	7
2.4.1	Regex Normalisation of Maskable Tokens	7
2.4.2	Fused Word Splitting	8
2.4.3	Spell-Checker Based Normalization	8
2.4.4	Hashtag Recognition and Substitution	8
2.4.5	Excess Letter Reduction	9
2.4.6	Abbreviation and Slang Mapping	9
2.4.7	Excess Punctuation and Emojii Deletion	9
2.5	Existing Research	10
2.5.1	Banerjee et al, 2012	10
2.5.2	Han, Baldwin 2011	11
2.5.3	Kauffman, Kalita 2010	12
2.6	Translation Systems	13
2.6.1	Google Translate	13
2.6.2	Microsoft Translator	13
2.7	Analysis of Effectiveness	14
2.7.1	BLEU	14
2.7.2	Translation Edit Rate (TER)	14
2.8	Dictionary	15
2.8.1	PyEnchant	15

2.9	Programming Language	15
2.9.1	Python	15
3	Requirements and Analysis	16
3.1	Aims and Objectives	16
3.1.1	Pre-Processing	16
3.1.2	Normalization	16
3.1.3	Translation	17
3.1.4	Evaluation	17
3.2	Constraints	17
3.3	Requirements	18
3.4	Testing	20
4	Design, Settings and Methods	21
4.1	Project Structure	21
4.1.1	Text Files	21
4.1.2	Extraction Files	22
4.1.3	Normalization File	22
4.1.4	Evaluation File	23
4.1.5	GUI File	23
4.2	Implementation	23
4.2.1	Text Files	23
4.2.2	Extraction	26
4.2.3	GUI	26
4.3	Methods & Techniques	28
4.3.1	Hashtag Normalization	29
4.3.2	Account Tag Normalization	29
4.3.3	URL Normalization	30
4.3.4	'Fused Word' Normalization	30
4.3.5	Excess Letter Normalization	32
4.3.6	Other Regex Normalizations	33

4.3.7	Slang Normalization	34
4.3.8	Date & Time Normalization	35
4.3.9	Spellcheck Normalization	35
4.4	Translation System	37
4.5	Evaluation Metrics	38
5	Testing & Evaluation	39
5.1	Introduction	39
5.2	Non Evaluative Testing	39
5.3	Experiment 1 - System Testing	40
5.3.1	User Acceptance Testing	40
5.3.2	Set Information	41
5.3.3	Results	42
5.3.4	Discussion	42
5.4	Experiment 2 - Development Tweets	42
5.4.1	Set Information	43
5.4.2	Results	45
5.4.3	Discussion	45
5.5	Experiment 3 - Parallel - Gold	46
5.5.1	Set Information	46
5.5.2	Results	47
5.5.3	Discussion	47
5.6	Experiment 4 - Parallel - Noisy	48
5.6.1	Set Information	48
5.6.2	Results	49
5.6.3	Discussion	49
6	Challenges	51
6.1	Slang Words	51
6.1.1	In Vocabulary Slang	51
6.1.2	Out Of Vocabulary Slang	51

6.2	Proper Nouns	51
6.3	Punctuation	52
6.4	SMT Systems	52
6.5	Reference Translations	52
6.6	Character Encoding	53
7	Conclusion	54
7.1	Aims & Objectives	54
7.2	Successes & Failures	54
7.2.1	Successes	54
7.2.2	Failures	55
7.3	Further Work	56
8	References	57

1 Introduction

1.1 Introduction to Twitter

First launched in 2006, Twitter is a social networking service that allows users to post 140 character blocks of text called 'tweets' that are then available to be viewed by anyone else in the world (depending on privacy settings). After rapidly gaining popularity, Twitter is now one of the top ten most visited sites in the world [10], with over 140 million active users in 2012[11], the company has defined itself as one of the foremost leaders in social networking.

One of the main attractions of twitter is that anyone can follow anyone else who has an account, from news outlets to famous celebrities by one click of a button you can receive up to date tweets into your own personal timeline. A big feature of Twitter is that the following of an account does not mean they have to follow you back, allowing large 'twitter-famous' accounts to generate social media content for thousands of followers.

Tweets present a very important data source for many international companies. Twitter can be used to build a web presence, to increase customer service and to advertise products both directly to the user via 'sponsored' tweets that appear in a users news feed. A foreign company looking to expand into a new country might use twitter to build a collection of tweets from that target nation and then utilize machine translation to collect customer sentiment.

1.2 The Problem

Since Twitter is used across all generations, the majority being 18-29[12], we see an ever changing language of 'Twitter English'. The character limit imposed by the nature of Twitter means we still encounter abbreviated forms of words common to SMS messaging, many will be familiar with abbreviation slang like 'LOL' for 'laugh out loud' or 'OMG' for "Oh my God!". The youth culture present on Twitter is constantly churning out new slang terms that spread rapidly between regions and countries, requiring constant observation and understanding of what these new terms mean. The use of punctuation to construct emoticons as well as the ever growing dictionary of emojiis [20] presents a struggle for machine translation systems as even though an emoji such as a diamond may have a one to one translation the translation system generally has no way to understand this.

The combination of irregular language and 'Out Of Vocabulary' (OOV) terms present a big

problem for machine translation systems due to the extreme amount of noise and the inability in a lot of cases to simply make one to one swaps with language. A system looking at normalizing irregular tweets, that is tweets which contain at least one OOV term, must come up with an approach or series of techniques to make the tweet easier to convert to another language.

@Aaron23 Yo man, you see the game last night?
#Liverpool win once again, #nothingcanstopus
glad i got 2 c ma sis 2nyt :)
Go to www.coolink.com to get the best deals!
I am a #winner and I know it.
Whatchusaayin! RT @kaz123 I shoot more hoops than Jordan bro #baller

Table 1: Example Tweets

1.3 Summary

It is the intention of this report to provide you with an firstly a detailed background on what will constitute the project with in-depth explanations of design considerations and terminology. Following that review I will talk about the implementation of the project along with results and any evaluations and discussions that arise from it's undertaking.

2 Literature Review

This section of the paper will document the problem analysis, supplemented with research of external papers. Initially I will discuss what development data is currently at my disposal before leading on to how to classify OOV tokens in that data. Secondly, the relevant techniques will be discussed before leading on to existing research in the project area. Finally, supplementary tools and resources relevant to the project will be explained.

2.1 Analysis of a 'Tweet'

Twitter is a social media platform from where users generate content in the form of "tweets" which are 140 characters worth of text. The 'tweet' contains some unique characteristics that separate them from other character constrained forms of communication, namely SMS text. The main features that make tweets unique is the use of 'hashtags' and 'mentions'. Hashtags, strings prefixed with a '#' are used to indicate a topic and other users will be able to find all tweets relating to the hashtag. Mentions are how users communicate direct to one other in the public view, writing "@Daniel" within the tweet will notify the account with username 'Daniel' that you have delivered your tweet.

In addition to this, tweets also include an extended range of language, informative strings of text such as URL's, emails and such are able to be easily copied and pasted into messages. The increased use of these means that the character limit on tweets is further strained and leads users to shorten words down in order to fit the character limit, similar to the way words were often abbreviated in SMS slang (e.g. 'OMG' - "oh my god", 'LOL' - "Laughing out loud").

Another important feature of tweets is the use of in-browser spell checking. In the past SMS messages would generally contain un-intentional errors mostly down to the user's own grammatical knowledge and the lack of a synchronous spell checking. However modern technology means that if users spell a word wrong it is instantly highlighted in red which generally increase quality of spelling across the tweet. This factor also means that for the most part, spelling mistakes by the user are intentional to either save space, convey different meaning or to express emotion.

A final point to made about tweets is the wide use in which tweets can be made about. A tweet has a wide variety of purposes not limited to, one to one communication, general opinions, quotes and news headlines. In particular twitter is a big platform for breaking news, for major events

many news companies operate twitter accounts with live feeds. These tweets generally contain a lot of proper nouns and therefore there is a problem presented in terms of differentiating words that are simply spelled wrong to these proper nouns.

2.2 Data Sources

For this project to be successfully undertaken we are required to use a collection of tweets firstly so that we can build up knowledge of 'twitter language' to help supplement a standard English dictionary. It is essential that we use or create a parallel set of tweets in English and one other language. A description of the data collections to be used are as follows:

- A previous MSc student in the Computer Science department obtained an extremely large collection of English tweets 12GB of plain text. On first outlook a text file of this size can not even be opened in any software that will be used and as such would be practically unusable if we were to operate on the whole data set. What makes this data-set most useful however is the fact that we have a large amount of useful tokens within the data set allowing us to have a solid base for developing our normalization techniques. For this reason we can designate this collection as our **development set**. Due to the immense size, it is also to break this collection into various sub sets to aid development.
- For our **test set** we will be using a large publicly available parallel data set[21]. The data set was obtained by using a web crawler to collect tweets that were tweeted in two languages. While this collection is extremely large and provides a useful basis for testing our techniques there are drawbacks. A big drawback is these tweets are not necessarily standardized in the parallel language and we will be dealing with a potentially poor standard of translation. Another drawback is that the translation is still in the form of a tweet as opposed to a standardized sentence devoid of tokens like emoticons.
- From the same source as previously [21] we can obtain multiple collections of gold standard parallel data. A gold standard parallel consists of an original source of tweets with a corresponding high quality human translation of each tweet. While of a high standard of translation this data set is ultimately not as useful to us as it would seem, mainly because

there are not a lot of tweets to choose from, nor a lot of tokens that need normalizing. We can however still use this data set to supplement the testing.

For a single data set in use over the project we would ideally have a data set that is:

1. Large in Size - So that we can find as many tokens as possible to confirm the effectiveness of each relevant normalization technique
2. Has an accurate translation - For evaluation of effectiveness in the context of machine translation this would allow us to "grade" the techniques produced and provide statistical evidence.

Because of the difficulty in finding a collection that satisfies these two attributes we will use multiple collections but use them in different contexts.

2.3 Classifying Tokens

To normalize tweets it is important that we break down each tweet into tokens and to determine a classification for each token and once that has been performed we can apply our normalization techniques and perform replacements.

Tokens that can be observed and some of which one of the main papers examined for this project [1] provides can be described:

1. Maskable Tokens - These tokens such as URLs, IP Addresses, Emails etc. Are tokens that will not be translatable as they are still relevant in any other language. The detection of these words using a regular expression means we can substitute in a placeholder that will not be translated when using the translation API. Issues with this token include easiness in which mistakes can be made when writing and by extension the large amount of regular expressions that will be needed to cover each instance of a maskable token.
2. Fused Words - These tokens entail instances where two or more strings are combined by a period symbol '.'. We must detect these situations, where potentially more than one period symbol is used, using a regular expression. While for the most part we want to classify instances where two valid words (checked against a dictionary) are joined, we will also encounter instances where two strings that are not part of the dictionary are also joined.

These instances may be the result of an error by the user e.g. "Hope liverpool become league win.ners". Here, win.ners should simply be winners and it is important that we look at these tokens and try and workout if they should simply be one whole word. An issue with this is the overlap between maskable tokens not detected by the system but we can account for this by using a list of known file extensions and addresses when we implement normalization.

3. Mentions - Mentions as talked about previously are account names on twitter that are prefixed with an '@' symbol. When sending a tweet the account 'mentioned' in the tweet receives notification that the tweet has been posted. Often we will find this token at the beginning of the tweet, for example "@DannyBoy Can't believe we won today". However mentions may appear at any point in the tweet and can make sense as part of the text e.g. "Taking @JennyGirl to the match today!" or "@JennyGirl is my girlfriend!". Therefore it is important that we can recognize that mentions may take the form of proper nouns.
4. Hashtags - Hashtags, in the context of tweeting, are any unpunctuated strings preceded by a '#' symbol. In most situations they are topical and relate to a function in twitter whereby clicking on any hashtag gives the user a list of other tweets containing the same hashtag. Hashtags have many uses and because they are mainly placed at the end of the tweet it initially seems like a good idea to simply ignore them. However, in tweets often people might hashtag a noun/word/adjective within the tweet but still intend it to mean part of the sentence. For example, "I hope #LFC go for the #win tonight.". In this example the tweeter uses the word win as both a verb and a hashtag so we have to be able to detect these instances specially. In addition, we might often find invalid hashtags for example #winn.ers or #can't where even though Twitter will not detect these as valid tags, within our system we must still detect them as hashtags and then further deduce whether they constitute a fused word or different.
5. Spelling Errors - These tokens are common across any form of language but it is specifically problematic within basic text processing as there is a high chance that we will correct the word to something not relevant to the sentence without the use of a context-sensitive spell checker (<https://github.com/mattalcock/blog/blob/master/2012/12/5/python-spell-checker.rst>). In addition to using a context-sensitive spell checker we can supplement our dictionary with

technical words that can be found in publicly available corpuses.

6. Slang Words - Words that are not encountered within a standard English dictionary but are used interchangeably in every day speech often come under "slang". Slang words often include abbreviations such as LOL which is a way of indicating laughter in text, in the context of text communication between people it is quite hard to translate abbreviations such as that as they cannot be replaced by a single term. Alternatively, a lot of slang encountered these days involves words that have a double meaning in the English language, for example the word "peak" in modern English slang is often used to express dismay at a negative situation, its use might involve saying "I lost my keys last night, it's so peak". The difficulty in this situation is that peak has an ambiguous translation into normalized English speaking, it could be interchanged with words such as "annoying" or "unlucky". Furthermore, the word peak itself has a translation into english as meaning the top of a mountain, or the maximum of some kind of quality ("the peak of Ben Nevis", "in peak condition").
7. Non-Translatable Words - While hashtags are a special case where sometimes we might obtain a different token by stripping the hashtag, we also have tokens such as account names (words proceeded by an '@' symbol) and usernames (aca13ab) which do not need to be translated.

2.4 Normalisation Techniques

The main focus of this project is to implement similar techniques described in a Dublin City University paper [1], as well as implementing further techniques to improve machine translation of English tweets containing out-of-vocabulary words.

2.4.1 Regex Normalisation of Maskable Tokens

Given that we can utilize regular expressions to classify tokens we can group the expression into parts. This method allows us to differentiate between something like a web address or file extension (pizza.co.uk, pizza.zip) and a joined word (pi.zza). We can accomplish this by generating or acquiring a list of known web extensions and file extensions and using this to check the ending

of a detected token to see if we can match it against a file type. Once we have indication the token can be masked we simply insert a place holder that will not be processed in machine translation. It is in our favor that most common file types generally do not appear as suffixes very often and thus errors regarding overlap between maskable tokens and fused words should be relatively small.

2.4.2 Fused Word Splitting

When we find Fused Word tokens where each string joined by a period, '.', belongs to the dictionary then we simply remove the period symbol and separate the words. However in cases where one or more of the joined constituents does not belong to a dictionary we can check that the combination of all constituent strings belongs to the dictionary. If so we replace the fused strings with that word.

2.4.3 Spell-Checker Based Normalization

One of the more common tokens found in tweets is that of spelling mistakes. This is mainly because place names, personal names and technical vocabulary are not present within standard dictionaries. One of the solutions to this is to implement a spell-checking that has been augmented using publicly available corpuses. The idea being using a supplementary corpus is that it would be possible to flag domain specific language. If the token we are examining is still an erroneous spelling we can then use a spell-checker to choose the highest ranked word that the spell checker suggests. In addition we could implement an advanced spell checker that uses probability theory similar to the way Google implements spell check suggestions [2] rather than simply looking at the most similar word.

2.4.4 Hashtag Recognition and Substitution

As mentioned hash tags might often be used as part of a sentence and it is important we recognize when this takes place. One of the methods of doing this is ignoring all hashtag tokens that are in a sequence present at the end of the tweet and incorporating all other hashtags into the sentence by removing the hash. For example "Hope #Liverpool become #winners tonight #FACup #Soccer" would become "Hope Liverpool become winners tonight #FACup #Soccer". This way even though 'Liverpool' is still an OOV token it may be picked up as a location name in the spell checker and

the word 'winners' will be in in vocabulary token and be fit for translation.

2.4.5 Excess Letter Reduction

A technique implemented in a widely cited paper, (Han, Baldwin 2011), involves words that have been deliberately misspelled by the user to convey extra meaning or to sound like one would speak it in real life given the context. Common examples include "Sooooon", "Noooooo" and "looooool" which should all really be spelled as soon, no and lol respectively. While the English language ordinarily does not allow words to have tripled letters there are a very few cases [4]. To normalize these instances we can either remove letters until we have an in-dictionary value or we can reduce all letters repeated down to an integer value.

2.4.6 Abbreviation and Slang Mapping

To adjust for the amount of abbreviations and common slang use in tweets that are a few approaches including using a third party website to find definitions [5]. Another is to build our own mapping of slang terms and abbreviations to regular speech. Utilizing work produced by a previous student involving analysis of OOV words in the 12GB selection of English tweets this project is using we can produce a mapping of words that have a direct translation into English.

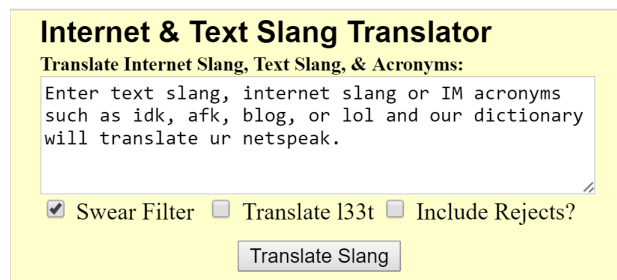
The image shows a web interface for an "Internet & Text Slang Translator". The title is "Internet & Text Slang Translator" in bold. Below it is the subtitle "Translate Internet Slang, Text Slang, & Acronyms:". There is a text input area with the placeholder text: "Enter text slang, internet slang or IM acronyms such as idk, afk, blog, or lol and our dictionary will translate ur netspeak." Below the input area are three checkboxes: "Swear Filter" (checked), "Translate l33t" (unchecked), and "Include Rejects?" (unchecked). At the bottom is a button labeled "Translate Slang".

Figure 1: Slang Translation

2.4.7 Excess Punctuation and Emojii Deletion

If we were dealing with sentiment analysis of tweets it might be a good idea to keep excessive punctuation ("COME ON YOU REDS!!!!") or emojiis ("So glad we won! :)"). In the first case

the excessive use of exclamation marks indicates excitement and in the second case the 'smiley' emoticon indicates happiness. However, in the context of machine translation it is best we keep the punctuation to a minimum and delete emoji's as they potentially have no meaning after the tweet goes through the machine translation system and only serve to cause confusion.

2.5 Existing Research

2.5.1 Banerjee et al, 2012

The work by (Banerjee et al, 2012) has provided a large amount of inspiration for this project. In particular it's use of rule-based normalization to alter forum data across languages.

Utilizing statistical machine translation tools, the purpose of this paper was to reduce the amount of OOV's across content with the hope that the SMT system would achieve a better translation. One of the main differences between (Banerjee et al, 2012) and this project is the source of data. In the original paper, Symantec forum data from across multiple languages was used to train a language model whereas in this project, we will not be building a model. This is a significant difference in source data, the forum data of a Symantec data contains a lot more focused terminology of symantec products as opposed to the complete random variety of terminology used on Twitter. Additionally, tweet-specific items such as hashtags and account tags are unlikely to be present in the original paper.

Despite the large difference in sources of data, the techniques implemented in (Banerjee et al, 2012) do still have a huge usage in the context of translating twitter data. While the original paper implements a series of regular expressions to capture memory addresses, registry entries etc. this sort of data is rare to find on Twitter. The idea of implementing an adapted spell checker that utilizes a wide range of corpus's would be especially important in the context of tweets.

One feature of (Banerjee et al, 2012) that in future could be implemented by a similar project is the use of sentence pairs to handle un-translated OOV tokens. However, the difficulty in accomplishing this task is firstly acquiring a good quality set of training data and this is perhaps beyond the resources available at the present.

2.5.2 Han, Baldwin 2011

While some of the techniques listed above do borrow from (Han & Baldwin, 2011), namely the removal of excess punctuation and letters, the approach by (Han & Baldwin) has some marked differences to the approach of this project.

The approach has two main rules, firstly, that the only tokens taken into consideration by the system are OOV words, that is, words that are not within the vocabulary being examined. Secondly, the system will only replace words that might have a single replacement, for example, 'smokin' to 'smoking' but not 'OMG' to "Oh my God". The vocabulary referred to by the system is that of the GNU 'aspell' dictionary, a dictionary that does not include proper nouns such as 'Wikileaks' or neologisms like 'hopeable'.

For the normalization of text, the strategy is to pre-process some of the words in the tweet and generating a confusion set of IV (in-vocabulary) words to replace any OOV words. This confusion set is based upon metrics of character edit distance such as in spell checking but also upon the edit distance under phonetic transcription via use of the double metaphone algorithm (Phillips 2000). Since the confusion set is large the top 10% candidates are selected based upon a generate language model of 100% IV tweets.

Instead of implementing various techniques to bring the tweet closer to it's normalized English translation the paper looks at predicting whether the word being looked at is ill-formed. This is based upon dependencies extracted using the Stanford parse (Klein and Manning, 2003; de Marneffe et al., 2006) on the New York Times corpus.

Since the normalization is based upon swapping an OOV token with a word from the IV Confusion set for that token, a variety of metrics are combined to rank each candidate with the most likely one chosen to replace the token.

One of the important techniques to be observed in this approach is the use of the double metaphone algorithm (Phillips 2000) to provide candidate replacement words for tokens such as '2geva' for together. (Han & Baldwin) present a unique approach that is may provide inspiration further down the pipeline for expansion upon existing techniques.

2.5.3 Kauffman, Kalita 2010

The approach demonstrated by (Kauffman, Kalita 2010) has a lot more overlap with this project than previously seen in (Han, Baldwin 2011). The paper produced by (Han, Baldwin 2011) primarily looks at normalization of tweets but not within the context of machine translation.

An interesting strategy of (Kauffman, Kalita 2010) was to use a previously published paper [8] to parse a list of SMS Words/English table. This aims to help avoid the error of a spell checker incorrectly classifying a 'slang' or novel word into a different meaning term. Once the table is generated any SMS terms that map to multiple english terms, an example given by Kauffman is wt -j, what,with, are discarded from the list of words we can map to regular English.

An additional technique that is worth noting is the idea that since most spelling errors on twitter tend to be a case of a single letter out of place and so for each OOV word if a swap of two adjacent letters causes the the string to match to the dictionary then we can assume that that is the word the user intended to write.

An overlap between (Han, Baldwin 2011) and (Kauffman, Kalita 2010) is that of the excess character removal, however both papers have different approaches. Whereas (Han, Baldwin 2011) looks at simply removing all excess letters down to three, and then substituting the word for its closest match in the confusion set, (Kauffman, Kalita 2010) simply cuts down the repeated letters until a match in the dictionary is found. A disadvantage of the approach by (Kauffman, Kalita 2010) is confusion between words with double repeated letter and words with a single letter on it's own. A good example of this is the adjusted word 'goooooooooooooood', if we remove the repetitions of 'o' we come down to the words 'good' and 'god'. (Han, Baldwin 2011) Will have more chance of classifying this as either 'good' or 'god' correctly as it will look at dependencies within the sentence whereas (Kauffman, Kalita 2010) will simply change the word to 'good'.

A further overlap and a benefit of the implementation of (Kauffman, Kalita 2010) over (Han, Baldwin 2011) is the incorporation of hashtags (#topic) and mentions (@username) into the syntactic analysis of the tweet. As mentioned in the hashtag recognition technique above we can make deductions over whether a hashtag has syntactic value depending on its position and surrounding tokens. (Kauffman, Kalita 2010) presumes that hashtags that are proceeded or succeeded by another hasthag or hashtags at the front or end of the tweet, are all topical and have no value with regards to the context of the tweet. It would be in the interest of this project to examine

this and it would be interesting to see how alternative presumptions would work, for example only hashtags at the end of tweets are presumed to be topical.

While (Kauffman, Kalita 2010) does look towards machine translation of normalized tweets there is a big difference between the direction of translation. (Kauffman, Kalita 2010) Looks at the use of a trainable translator [9] that translates the initial tweet and the normalized tweet into English. This is not of much use in the context of translation into a foreign language but it does demonstrate how we can obtain better normalization of a tweet using machine translation software.

2.6 Translation Systems

To accomplish machine translation of our lists of tweets we require an actual system to utilize. To avoid complication the project will use a publicly available API to accomplish the task of translation, the two main candidates being Google Translate and Microsoft Translator.

2.6.1 Google Translate

Google Translate, in it's statistical model form, was launched in 2007[15] and requires a \$20 payment per one million characters translated using the API[17]. While there have been studies on it's effectiveness[16], there is not a definitive answer as to what public API is the most effective. It is unfortunate that Google requires a payment of sorts as it would have been useful for the project to be able to compare both Google and Bing API's for the purpose of testing.

2.6.2 Microsoft Translator

Launched in 2009[18], Microsoft Translator offers a free service up to a limit where users are limited to 2,000,000 characters per month inputted into it's translator before paying extra. This factor makes it important to control testing so to not incur unintended financial costs. Unfortunately, availability of Microsoft Translator analysis is sparse and there is not enough material to make it informed comparison to Google Translate. Therefore, the deciding factor in choosing between Google and Bing is the cost, and hence Microsoft Translator would be the ideal third party translator that the project will use.

2.7 Analysis of Effectiveness

As mentioned previously, a metric is needed to compare translations of our original and normalized tweets. It would be useful for the project to examine multiple means of evaluation but at least one of the following metrics will be used.

2.7.1 BLEU

The BLEU metric [13] has unsurprisingly a core idea in that 'The closer a machine translation is to a human translation, the better it is'. Using a 'word-error rate', the BLEU metric uses a weighted average of variable length phrase matches against the reference translation that allows a leniency towards differently phrased sentences. BLEU is one of the most popular evaluation metrics in the field at present and can be seen used in (Han, Baldwin 2011), (Banerjee et al 2012) and (Kauffman 2011). For this reason, it seems fit that it will be one of the main metrics for evaluation of this project.

While the BLEU metric is popular it does have some drawbacks, especially when considered against the objects of evaluation in this project. Firstly, to compare a translation against a human translation the BLEU method is meant to use multiple reference translations. In the case of this project that is difficult as it is already difficult to obtain a single good reference translation, let alone multiple. Secondly, BLEU penalizes brief translations. By nature and its character limits, tweets are 'brief' and for this reason BLEU will penalize the translations of our source tweets.

2.7.2 Translation Edit Rate (TER)

Translation Edit Rate [14] is an alternative to BLEU that appears correlates just as well with the BLEU metric in regards human judgment of machine translation quality. Between a reference and hypothesis pair of sentences TER works out the number of edits required to edit the sentences to match divided by the average number of terms in the reference. Despite not being as popular as BLEU, TER offers an alternate metric and would be beneficial to implement in the project for a more complete evaluation of the system.

Since TER is concerned with the 'edits' required, the lower a TER score the better it is for the system. Especially for strings that are short, TER is less punishing on brief strings than the BLEU metric.

2.8 Dictionary

2.8.1 PyEnchant

As mentioned earlier, one of the normalization techniques simply involves using a spell checker as a last resort to attempt to bring an OOV token into an IV word. The primary reason for using the PyEnchant [19] dictionary is it's simplicity and effectiveness as well as the fact we can supply a standard En-US dictionary with a list of words to bring into vocabulary.

2.9 Programming Language

2.9.1 Python

All programming requirements for this project will be implemented using the high-level programming language Python. The reasons for using this language come down to personal fluency in coding in that language as well as it's lightweight scripting style which will benefit the approach of this project.

3 Requirements and Analysis

3.1 Aims and Objectives

It is the objective of this project to implement a set of normalization techniques on a collection of non-normalized, 'Twitter English' tweets and translating the tweets into a foreign language with the aim of achieving a better translation than if the tweets were non-normalized. The approach can be divided into four distinct stages which can be outlined below.

3.1.1 Pre-Processing

Given a large data set of tweets it is important both for the efficiency of the development process and the quality of implementation that we analyze our development set and selectively extract tweets which will be positively affected by the normalization techniques. This step involves producing regular expressions and methods to identify relevant OOV tokens that will be beneficial for our system to normalize. By making a customization system we can extract high quality sets of tweets to be operated on by each relevant normalization technique. If we were to implement a technique that for example dealt with the normalization of hashtags then we would be able to extract X number of tweets from the development set that use hash tags.

3.1.2 Normalization

Our next step is to develop various normalization techniques over a large English only set of tweets before using them on a test set of which we can then use the results to evaluate the techniques. The techniques will initially correspond to implementations described by (Banerjee et al 2012) but using further research and personal innovation it is within the scope of the project to utilize any additional techniques that can be found and implemented. By implementing normalization techniques on a set of tweets extracted from our larger development set the project will aim to develop each technique as much as possible. This can be aided by generating printouts of the tweet itself followed by the normalized version, with outputs of each instance of normalization to aid development and to highlight potential cases where normalization does not appear to benefit.

3.1.3 Translation

Once at a stage where a satisfactory number of techniques are implemented we can then start looking at using these on a test set on either parallel tweets (that is tweets in one language and a foreign language but maintaining the same 'tweet' style) or a gold standard data set of non-normalized tweets in English with a corresponding human translation into standard English. Putting the test set through the system should generate both a list of original english tweets as well as a list of normalized tweets. Incorporating API methods of a public translation system (Google, Bing) we can machine translate each of our tweet files into our desired language giving us two more lists of tweets.

3.1.4 Evaluation

To evaluate the effects of our system we must find some way to compare our two lists of translated tweets to the parallel translation within our test sets. There are multiple available metrics for this including but not limited to BLEU[13], TER[14] and string edit distance. By running both the translated tweets and the proper translation through the system we can automatically calculate the evaluation metric for each tweet and calculate the mean value across the whole of each file respectively. The project hopes to record an increase in value of the evaluation metric for the normalized tweet translation to the non-normalized tweet translation. Results will be tabulated and multiple test can be run to measure the effectiveness on each normalization technique along with the overall effect.

3.2 Constraints

While there are not many constraints within the project there is a particularly large problem to do with the quality of translated tweets that we will use to compare to our machine translated original and normalized tweets. While we are able to obtain a large dataset of parallel tweets, the quality of the parallel can not be assured for quality compared to say a smaller dataset of gold standard human translations. This has the potential to affect results, for example the parallel translation may not actually be a good translation at all and as such may have a more randomized effect on the value of evaluation metric for each translated tweet.

The natural solution to this would simply to use a gold standard dataset of original tweets

and a high quality translation of said tweet, however, the difficulty with this is that gold standard datasets are very time consuming to collect. The ability to procure a dataset from elsewhere would most likely require payment of sort or at least the willingness of bi-lingual volunteers to translate the original tweet to their second language. While some gold standard datasets may be obtained online, the sample size is likely to be quite small and not the size we need to properly evaluate the system.

A further constraint of the system is the use of third-party translation software. Public API's such as Google and Bing require a payment plan for unlimited use of translation, although Bing does offer a free service up to a limit. Given that amount of translation that will be performed the cost factor of the translation systems may require downsizing the testing.

3.3 Requirements

A list of requirements for each part of the system may be found below.

Priority:

M - Mandatory

P - Priority

O - Optional

Pre-Processing	Priority
Take an input file of tweets and read the file line by line.	M
Evaluate and calculate the number of OOV tokens within each line.	M
Sort the input file by each category of token, dependent upon parameters.	M
Extract a parameter value of tweets and output it as a new file.	M

Table 2: Functional Requirements

Normalization	Priority
Take an input file of tweets and read the file line by line.	M
Evaluate each OOV token and determine a normalization technique to apply.	M
Selectively apply normalization techniques based upon parameters.	P
Implement all normalization techniques listed in the literature review.	M
Output a new file of normalized tweets.	M

Table 3: Normalization Requirements

Translation	Priority
Take two input files of original and normalized tweets.	M
Implement a public translation API.	M
Translate each line of each file, using the API, into a foreign language.	M
Output two new files of translated original tweets and translated normal tweets.	M

Table 4: Translation Requirements

Evaluation	Priority
Take three input files of machine translated original and normalized tweets and human translation of original tweets.	M
Implement the BLUE evaluation metric.	M
Implement the TER evaluation metric.	O
Calculate the mean average evaluation metrics for each file of machine translated tweets against the human translated tweets.	M
Output mean average values.	M

Table 5: Evaluation Requirements

Extra	Priority
Implement a graphical user interface combining all four systems.	P
Allow user input for manual testing of a single pair of original and standardized tweets.	O
Apply good coding practice.	P
Apply PEP 008 coding standards.	O

Table 6: Extra Requirements

3.4 Testing

Testing for this project will be completed in two phases:

1. The first phase of testing will consist of user acceptance testing. I will be using a self produced set of tweets that will be supplemented with edge cases provided by the large development set provided.
2. The second phase is inherent in the project goals. This will consist of the evaluation of the system as a whole using the two evaluation metrics, BLEU and TER.

Once the first phase of testing is completed, I will be able to focus on the second testing phase and work on improving the existing techniques and supplementing them with extra techniques that are have not already been described. The first phase will again be referred back to and supplemented, consistently checking that the normalization process produces the desired effects.

4 Design, Settings and Methods

This chapter will talk about the design of the system, this includes the structure of the programming itself, including the methodology for each technique. I will also talk about the settings and methods that were used to construct the system and to assist in accomplishing the project goals.

4.1 Project Structure

The figure below details the general pipeline of the project. Initially we have the research and literature review carried out earlier in this paper before leading on to the four main stages of the project. Finally we end with a presentation of the results found.

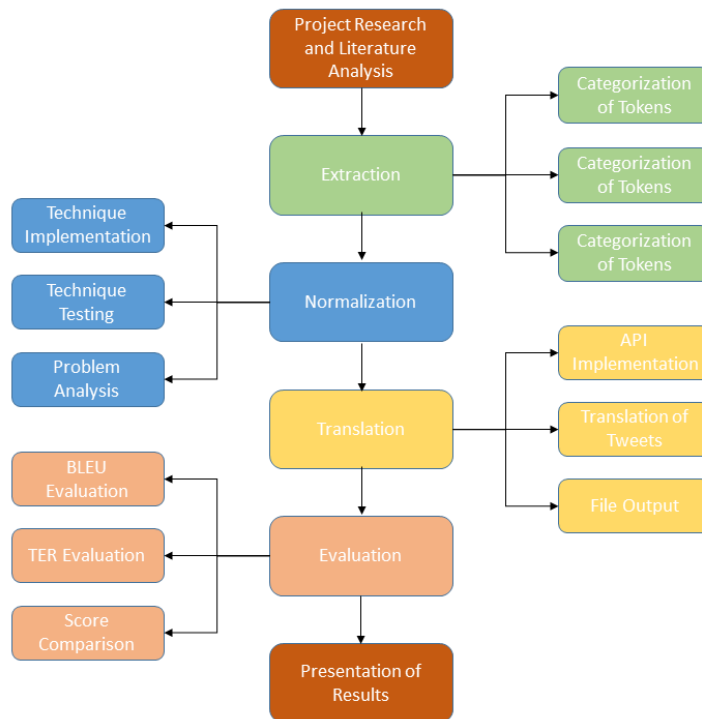


Figure 2: Project Plan

4.1.1 Text Files

The first step in undertaking this project is to design text files that will be able to be opened or outputted by the program. There are at least two simple ways in which a python program can parse text files that I think would be suitable for the project. The first way is a very simple line

by line text file, where each line of the file corresponds to a tweet that we will then assign to a variable which can be operated on.

The advantages of using a line by line approach is that the code for processing the document is very simple and relatively fast. However, this all means that the data contained within that text file is also required to be simplified and the text must also be treated the same. For example, if were to have any data other than just a file of tweets contained within the text file, we would have to load the text file into memory to find that ulterior data and this will just slow down the system massively. The components of the program which will utilize .txt files include both the input tweets that we will perform normalization on but also corpus's of non-standard English words which will improve spell-checking effectiveness.

To complement this simple line by line input method, python offers methods to parse JSON files into a python associative array as well as having methods to output into a neatly organized JSON file. Given that, the project will be able to firstly utilize a set of tweets that we can operate on line by line and then output both the original tweet and the normalized one into a JSON text file. Secondly, we can take this output JSON and input it back into the program along with translated versions to calculate scores for the quality of the translations.

4.1.2 Extraction Files

As mentioned before, the program will rely on having a text file that simply contains lines of real tweets. However, to allow for multiple data sets the program should be able to process large text files and be able to select a "range" of tweets. It can not be assumed that the input file of tweets will be completely random. Therefore to avoid discrepancies in testing the overall effectiveness of the program but to also provide exposure to further test cases there needs to be a way to break down larger files of tweets. An additional advantage of this is that we can automatically break down larger files to speed up the program but to also avoid manually doing it ourselves. This is especially useful when the source of the tweets is too big to open in text editors!

4.1.3 Normalization File

While there is the opportunity to separate each technique implemented into individual files, for the purposes of this project all the normalization techniques are written in one file.

The normalization file is programmed as a class as opposed to a module. A python module is generally just a file of methods that can then be imported and used normally. A python class on the other hand allows me to create a normalizer object. The reasons for making an object is that we can separate the program structure itself for clarity but also treat the object as the main part of the system. By that we mean it has it's own set of variables that can be accessed, we can then create a GUI that can then interact with the normalizer to display values and run code. These variables will consist of a selection of values, e.g. the number of occurrences for each OOV token in an input file.

4.1.4 Evaluation File

To keep the project organized we keep any code that involves creating metrics to a single evaluation file. This file contains methods that take input and can deliver an output from that, in this case, the TER and BLEU metrics. Unlike the normalizer, this file is a module and as such it's methods can be accessed without assigning a class object.

4.1.5 GUI File

While it is completely possible for the project to be undertaken and completed by simply using command line. I felt that, with the amount of testing needed and the amount of alterations to code, I would be able to save more time creating a GUI that going without.

The idea of this interface is that I can run code with different inputs as many times as wished, without restarting the program from command line. The implementation complements the use of a separate object that is used to normalize our input tweets.

4.2 Implementation

4.2.1 Text Files

As mentioned previously, the project program takes in input from text files. The first type regular lines of strings where each line corresponds to a a single type of information such as a tweet. The second type corresponds to a JSON formatted text file. There are three input files that we use that are of the first type:

1. Development Tweet Set - This file is a very large set of tweets that can not be opened in an editor (12+GB), It can however be processed without storing the whole set in memory. The development tweet set has presented some problems in the way it is accessed and used. One problem is that as the file is so large it is not possible to view in the editor as mentioned, this means that we must first choose a 'range' of tweets to extract. Secondly, from extraction it has been observed that the tweets are generally grouped by a topic or feature. For example the first few thousand tweets involve certain users that are generally tweeting about Wigan Athletic Football Club. The reason this presents a problem is that certain topics will have certain characteristics and from a testing perspective it means we are challenges with extra processing to construct a randomized set of data.
2. Default Tweet Set - Using manually created tweets we can produce our own specific data set that the program will use automatically each time. This consists of 40 or so lines only.
3. Default Corpus File - As part of the normalization process this we can supplement a dictionary with specific words which are not regular English. As opposed to the other files each line corresponds to a word only.

There are also three other text files that are needed as part of the program but that are in JSON format instead. Their format will be converted to the JSON format shown below.

```
{
  "data": {
    "id":
      "transOrig": --the original tweet translated--,
      "transNorm": --the normalized tweet translated--,
      "transPerf": --the perfect translation of the tweet--,
      "orig": --the original tweet--,
      "norm": --the normalized tweet--,
      "perf": --the perfectly normalized tweet--
    }
  }
```

Figure 3: JSON Data Format

1. Output JSON - This JSON is outputted by our program after inputting our tweets that we want to normalize. It contains only the original tweets and their normalized version with some empty fields corresponding to the translated versions of the tweets and also the "perfect" normalized version.
2. Input JSON - The input JSON is intended to be used to generate scores for each evaluated metric that we want to use. Before inputting this JSON we can manually edit in the translations for us to be able to evaluate each item in the JSON data object.
3. Test Tweet Set - A large file of 'parallel' tweets that have been found using a web crawler. They are conveniently provided in JSON format but to be able to interface with the project code they require some more processing to extract the JSON values correctly and to output them to the Output JSON file.

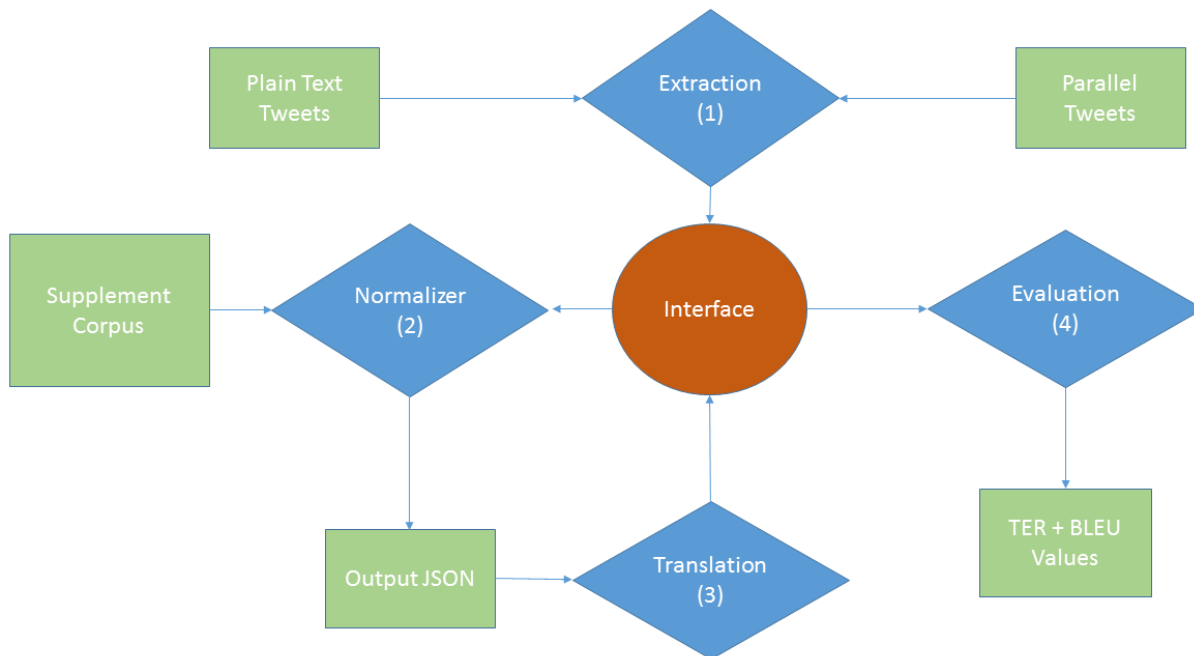


Figure 4: Program Structure

4.2.2 Extraction

The tweet extractor implemented takes a 'base' number and a 'max number, before outputting the tweets between those two numbers (where each tweet is separated by a line break. The output file is a text file of tweets in the manner of the input (as there is no other data given other than the tweet text).

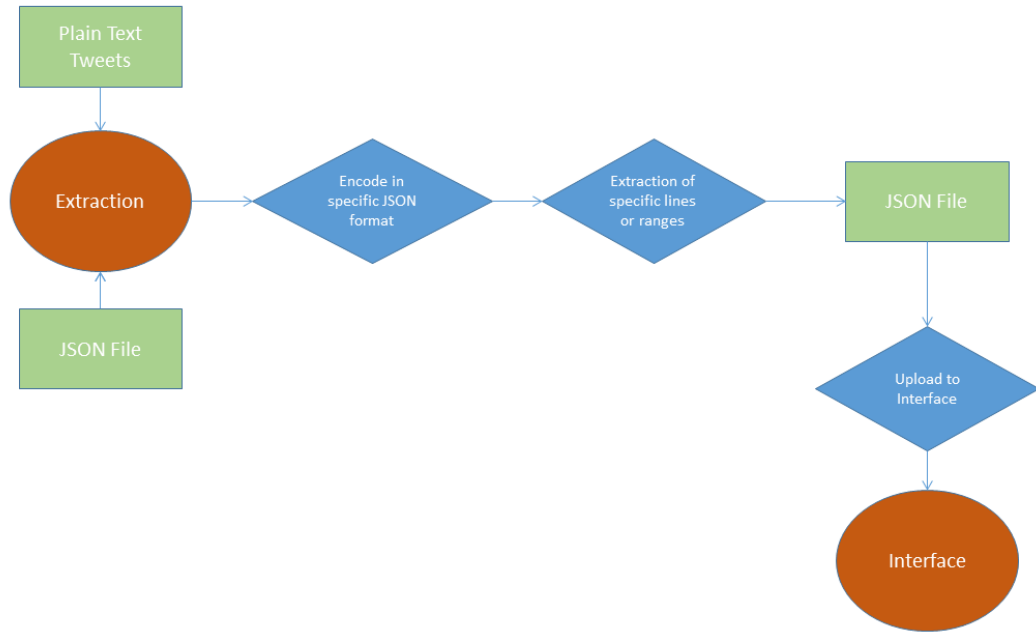


Figure 5: Extraction Structure

Inputting the parallel data files into the system was problematic to implement. By default python uses only ASCII encoding and therefore, inputting non-english characters was problematic. The solution was to use the 'codecs' library to specifically encode characters in 'utf-8' and store them in the JSON files correctly.

4.2.3 GUI

The GUI implemented for this project uses the Tkinter library[23]. As the project is a research project as opposed to a software development application for consumer, Tkinter was suitable for it's simplicity both in a programming sense and also visual. The GUI library works on a row and

column system and therefore each row generally corresponds to a function.

Figure 6: GUI

The first row has two functions. The first function on the left noted by the button "Run Code", runs the normalization code on a data set that can be uploaded using the "file button". The check buttons on the far right are used to select whether to handle the file input on a line basis or as a json, and also whether to add printouts to the program. The final function of that row is to extract from a file lines in a range defined by the two entry boxes: 'Min Line' and 'Max Line'.

The second row is concerned with which tweets we wish to be returned or printed out to the console. For example if we only wanted tweets containing hashtag to be returned we would select that box and de-select all. This feature of the GUI was primarily for the use of testing.

We can decide to use specific normalization techniques in the third row. By default they are all selected and ready to be used but if we wanted to perhaps test our system without using spellcheck we can de-select the spellcheck box.

Next, we have the automatic evaluation function. This is where we can input one of the formatted JSON files produced by the system. Once the file is uploaded pressing the 'evaluate' button the system will perform TER and BLEU evaluation metrics on each tweet. The resulting average from this will be displayed next to the labels for each metric.

Alternatively there is the manual evaluation. Primarily for user-acceptance testing, this function of the system also allows for demo usage. Entering a tweet in the far left entry box, we can click 'normalize' to provide a normalized version. Pressing 'translate' give us translated versions and finally 'evaluate' gives us BLEU and TER scores for the data constructed.

Finally, we can test out the spellcheck function of the system by entering a word into the entry box. Pressing check will return 'TRUE' if the word is in dictionary but if the return is 'FALSE' we recieve that label but also the best suggested replacement word.

4.3 Methods & Techniques

This section details the various techniques that were implemented, as well as including the pipeline of the normalization part of the system in image format below.

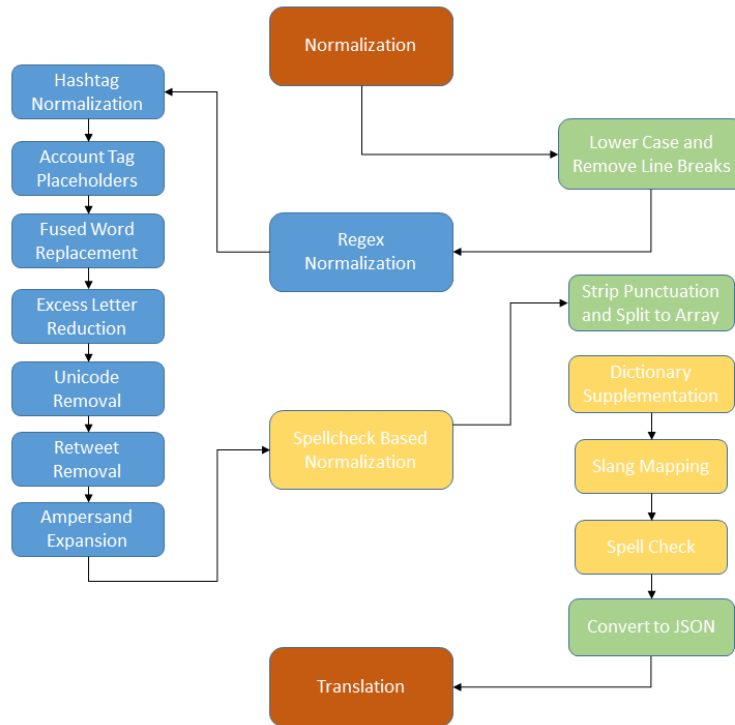
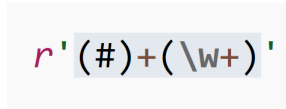


Figure 7: Normalization Structure

4.3.1 Hashtag Normalization

The first step in normalizing hashtags, and for any token we want to operate on, is to produce a regular expression. This allows us to capture a) the hashtag (or hashtags if the user has made a typing error and double the hashtags) and also b) the word contained within the hashtag.



```
r'(#)+(\w+)'
```

Figure 8: Hashtag Regular Expression

Firstly we want to check if the hash tag is the last token with the tweet. If it is, due to the nature of tweets, the hash tag will generally not be part of the tweets meaning and we can substitute it for a placeholder. If not, we can check if the following token is a hashtag, if it is, the hash tag is again not likely to be a meaningful word and we can replace it again. The final check is if the hashtag word is in our dictionary and we can then remove the hash if it is.

Original	Normalized
My word it's a #beautiful day #sun"	my word it's a beautiful day #sun ?
Hope #L'Pool win the football tonight!	hope Liverpool win the football tonight

Table 7: Hashtag Normalization

An issue when developing the normalization for hashtags was that of how to recognize when the hashtag is part of a sentence structure. A decision needed to be made of how far to go when normalizing the token and performing operations to normalize the hashtag contents itself using extra techniques. It was decided that in the interest of translation, the only time when the '#' of the hashtag is substituted would be when the hashtag text is in the dictionary. Examples can be seen where a hashtags contents include a series of words without spaces but trying to normalize this text leaves our system open to a higher degree of errors.

4.3.2 Account Tag Normalization

Account tags are used to a) reply to a user on Twitter or to b) notify them in a tweet. They can be used as part of a sentence or even to tag multiple users. Using the following regular expression,

we can capture each tag and replace it with a placeholder.

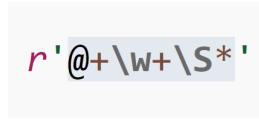


Figure 9: Account Tag Regular Expression

Initially it was set out to be able to just remove these account tags as often they were used at the start of a tweet and held no meaning. However, account tags are also often used instead of names and as such there were translation errors of a grammatical aspect. Therefore, a placeholder was used instead with the original tagged mapped back in post normalization.

4.3.3 URL Normalization

There are many possible regular expressions that we could use for recognizing URLs but because of the nature of the URL not all of them will be perfect. Similar to account tags, URLs may sometimes be used as part of a sentence. However, it is common to see tweets that contain a URL at the end of the tweet, generally relating to the tweets object of discussion.

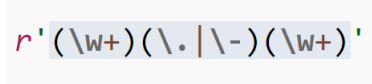
I experimented with a few regular expressions to use to capture the URL's of a tweet but found that there was one particular regex that worked the best out of those tried. This was mainly down to the fact that this regex in particular had been implemented with the purpose of finding url's in plain text as opposed to validating them. I eventually settled on a regular expression developed on GitHub[24].

A difficulty in normalizing URLs was that direct url's for say a file-path were difficult to construct effectively. Since the number of relative file paths contained within tweets is generally very low, this is only a minor issue.

4.3.4 'Fused Word' Normalization

When a word is 'fused' by a period symbol it is confusing from a perspective of punctuation. It is unknown as to whether the word is meant to be together to form a word and the period is there by mistake or whether the user is trying to fit the Twitter character limit by removing the grammatically correct whitespace. If at least one of the words before or after the period is not

present in our dictionary we can check to see if they combine to form an in-dictionary word. If there is, we replace the term matched by the regex with the new word. If not, we treat the regex as just grammatically incorrect and remove the period symbol.



The image shows a regular expression pattern: `r'(\w+)(\.||-)(\w+)'`. The pattern is highlighted in a light blue box. The `r'` is in red, and the rest of the pattern is in black.

Figure 10: Fused Word Regular Expression

The problem when encountering this instance of token is that there are sometimes words with multiple full stops, an example being ellipses. While ellipses can be recognized and dealt with as part of a piece of text, something like a word containing periods at three separate locations may be interpreted as a URL.

Original
cr.ude should turn into crude but clue.less won't be
No-one realizes how cool Mario is!

Table 8: Fused Word Normalization Examples 1/2

Normalized
crude should turn into crude but clue less won't be
no one realizes how cool Mario is

Table 9: Fused Word Normalization Examples 2/2

4.3.5 Excess Letter Normalization

There are numerous ways users can use more letters than needed in words for emphasis. The regular expression to capture these instances is simple and it is also simple to substitute each series of letters down to a number such as 2 (as there are very few instances of tripled letters as mentioned at 2.4.5).

A screenshot of a code editor showing a regular expression: `r'(.)\1{2,}'`. The opening quote is red, the closing quote is red, and the rest of the expression is in a light blue font.

Figure 11: Excess Letter Regular Expression

In practice it was found that when used in conjunction with a spellchecker reducing all instances of multiple letters down to a doubled letter, there was a higher degree of error observed. While intuitively the action would be to reduce to 2, to allow for words such as "too" to be correctly spelled instead of "to". When reducing ALL cases of multiple letters, the spellchecker would not return the correct word. Changing the number to 1 instead allows the spellchecker to have a greater chance of suggesting the correct word, even to words that have had their double lettering emphasized.

In the regex, we capture ANY character that is repeated more than once. This means it will include numbers and punctuation. Punctuation will be stripped anyway prior to spell checking operations and if the regex captured is a number we can skip the whole operation.

Original	Normalized
Hellloooooooooo everyone, love you all.	hello everyone love you all
Toooooo many people hate Bieber	to many people hate bieber

Table 10: Excess Letter Normalization Normalization

4.3.6 Other Regex Normalizations

This section of normalization techniques, I will mention techniques that affect only a very small or specific amount of words.

Firstly to deal with twitter specific language such as the manual insertion of 'RT' before re-tweeting we are presented with an issue of design and what we would classify as a 'normalized' tweet. We go about this by checking each token after it has been split by space for a match to 'rt'. For each case this is true the token is removed from the normalized text.

<code>r'\\brt\\b'</code>	<code>r'\\u[a-zA-Z0-9]{4,5}'</code>	<code>r'(^ \\s+)&(\\s+ \$)'</code>
RT	Unicode	And

Figure 12: Other Regular Expressions

In some of the test data the ampersand (&) sign was represented in the form of '&'. A simple regex was formed to find any ampersands or it's representation and to replace them with a simple 'and'.

In the development test set there were large numbers of unicode representations, many cases being emojiis that had been converted for testing purposes. Since the unicode tag nor it's representation on twitter would offer no meaning during translation, they are simply removed wherever they are found.

We do then have the problem of when tweets are used where the emoji or such is part of the object of discussion as can be seen below. In these cases there would be the opportunity to map unicode representations to a word but this would likely cause further bad translations in the cases where the emoji is simply emotive.

Original
RT @kmballoons displays delivered for the party
@dannyhulme7 I was floating around at the end&you know it!
Goodnight Twitter! /u2665Xxxxxx

Table 11: Other Normalization Examples 1/2

Normalized
@kmballoons displays delivered for the party
@dannyhulme7 I was floating around at the end and you know it
goodnight twitter

Table 12: Other Normalization Examples 2/2

4.3.7 Slang Normalization

When slang is discussed in the scope of the project we are only able to recognize slang that is not already in the dictionary. This includes words that have been deliberately shortened or words that contain numeration.

```
def initializeSlang(self):
    self.slang_dict["lol"] = "laugh out loud"
    self.slang_dict["g2g"] = "got to go"
    self.slang_dict["b4"] = "before"
    self.slang_dict["gr8"] = "great"
    self.slang_dict["2morrow"] = "tomorrow"
    self.slang_dict["2geva"] = "together"
    self.slang_dict["r"] = "are"
    self.slang_dict["u"] = "you"
    self.slang_dict["brill"] = "brilliant"
    self.slang_dict["vino"] = "wine"
    self.slang_dict["n"] = "and"
    self.slang_dict["ur"] = "your"
    self.slang_dict["every1"] = "everyone"
    self.slang_dict["omg"] = "oh my god"
```

Figure 13: Slang Dictionary

To normalize slang in a real-word application it would be necessary to create a very large dictionary of words and their replacements. For the purposes of this project, our dictionary of slang replacements can be compiled within the normalization file of the project. Instead of using

a regular expression, we would simply check each non-dictionary word against this dictionary and replace it as fitting.

Due to the large amount of variation on the way words can be shortened and spelled on twitter the current implementation of the project is only able to demonstrate normalization of slang on select items of text. While very small benefits of this technique may be observed in testing it is likely to not have any marginal effect at all overall.

4.3.8 Date & Time Normalization

Originally, the system was designed to be able to recognize various implementations of dates and times within tweets. A fairly complex regular expression would have been constructed and tested to capture these instances and to insert a place holder.

What I found during my implementation was that the Google translate system used to translate my tweets worked better without any attempts to normalize any dates or times. For this reason, attempts to normalize tokens that correspond to date or time were not carried forward with the final state of the project.

Original	Translated
Are you coming out on 20th Dec?	Vous venez sur le 20 dcembre ?
Mon 17th is when we're off to Paris	Lun 17 est lorsque nous sommes en route pour Paris

Table 13: Date Translation Examples

4.3.9 Spellcheck Normalization

A very important component of the text normalization is that of using a spellchecker to attempt to correct any words in the text that have not already been substituted by a method of normalization. The main spellchecker used for this task was the library mentioned previously, pyEnchant[19]. It's main advantages include offering multiple suggestions for incorrect words and being able to supplement the main dictionary with our own words. These suggestions are presented in the form of an array, where the best suggested replacement is the first element.

As mentioned, pyEnchant allows the supplementation of the main dictionary with that of our own words. The GUI made for the project includes an option to load a default file into the dictionary. Similar to that of the slang dictionary, this supplementary technique is proof of concept that it is possible for us to 'enhance' our spellchecker with that of any corpus's provided they are inputted into the default file mentioned in the correct way.

```
>>> import enchant
>>> d = enchant.Dict("en_US")
>>> d.check("Hello")
True
>>> d.check("HeLo")
False
>>> d.suggest("HeLo")
['He lo', 'He-lo', 'Hello', 'Helot', 'Help', 'Halo', 'Hell', 'Held', 'Helm', 'Hero', "He'll"]
```

Figure 14: PyEnchant

One of the ways that was used to supplement the spell checker dictionary was by using some research that was previously performed on the development test set of tweets. In this research there was an output file of OOV words, as mentioned before, words that were not found in the English dictionary. These words were reverse sorted by their frequency of occurrence. While a lot of these words were simply exaggerated spellings, or repeated letters, many of them were names of popular companies. By writing a script I was able to loop over the collection of words and extracting all the words that would be of use to the system and adding them to the corpus improved the spell checker.

The problems that came along with using the spellchecker were mainly matters of API usage and supplementing the default dictionary with my own terms. With regards to the corpus supplementation the current implementation within the project is a more proof of concept instance. The reason for this is that the normalization is performed on general topic datasets. To put together a corpus of names for the data set would be excessively time consuming with a very small impact on the final results.

1	retweet	110311
2	xxxxx	100659
3	xxxxxxx	56072
4	tumblr	53623
5	hashtag	46798
6	loooooo	44200
7	instagram	40807
8	tweeps	37577
9	t-shirt	35739
10	xxxxxxx	34725
11	*sigh*	34089
12	sooooo	33164
13	retweets	31437
14	no-one	29731
15	retweeted	26183
16	loooooo	25108
17	face*	24326
18	don't.	23784
19	xxxxxxxx	22719
20	unfollowers	21462
21	snapchat	19331
22	retweeting	18855
23	e-mail	18454
24	*hugs*	18385
25	checked-in	17329
26	whatsapp	16221
27	xxxxxxxxx	15347
28	soooooo	15160
29	loooooool	14982
30	and/or	14973
31	make-up	14311
32	line-up	14058
33	spotify	13822
34	nnnnnn	13190
35	t-shirts	12929
36	don't,	12595
37	nooooo	12384
38	*cough*	12156
39	m/s	12103

Figure 15: Previous Analysis

4.4 Translation System

The translation API that is used for the project is that of Microsoft Translate. By setting up an account online and implementing a python library I was able to construct a pipeline by which to automate the process of normalization, translation and evaluation.

The plan was originally to implement the more popular Google Translate API or atleast to use the translate system by hand. However, with a 20,000,000 free character translation limit offered by Microsoft, it proved beneficial to the system to use that.

While not integrated into the system application itself, there are reasons for this. Firstly, for the purposes of testing, different test files may need varying degrees of editing and secondly, the evaluation process uses different languages and it was easier to use command line tools to translate the data.

4.5 Evaluation Metrics

To implement the evaluation metrics BLEU and TER I was able to find some python libraries to include within my project. For the BLEU metric I can use the Natural Language Tool Kit (NLTK), which provides an API for which to compare translations. For TER there is the python package pyter that provides a single method for evaluating two single strings of text.

For the BLEU evaluation metric, there is the option to use a 'smoothing' function that allows us to evaluate translations on a sentence level. This sentence being each translated tweet and the corresponding translation. The smoothing function selected to be used was the method implemented by (Lin and Och, 2004).

Using these pre built libraries has allowed me to spend more time, working on the system as a whole.

5 Testing & Evaluation

5.1 Introduction

The testing of the system and to check whether it works as intended was performed via user acceptance testing. Unit testing was considered in the approach but as the project developed it was felt that user acceptance was the most economical method of testing to proceed with.

For the evaluation of the system there were multiple phases. One of the features of the evaluation was to look at the main techniques individually as well part of the system as a whole. The prioritized techniques were:

- **Regex Based Normalization** - All techniques that involved using a regular expression to capture tokens and then replace them according to rules.
- **Supplemented Spellchecker Normalization** - This includes using the spellchecker library `pyEnchant` along with a supplementary corpus of terms which can be expanded. In addition there is a dictionary of slang terms which provide replacements for each token post-normalization.

To get a varied evaluation of the system it was decided to use multiple experiments on different sets of data. These experiments will be explained in more detail in section 5.3.

The metrics both require an explanation of their score as both differ in what they mean.

- **TER** - The closer a score is to 0, the closer the string resembles it's reference string (translation).
- **BLEU** - The closer a score is to 1, the better the translation. Noting that a perfect translation does not necessarily give a score of 1.

5.2 Non Evaluative Testing

It should be mentioned that testing of auxiliary components of the overall system, e.g. Extraction, Comparison, were all tested firstly on an individual basis. Using command line tests I determined that each component first worked on mundane tests before then integrating into the system as a

whole. This integration consisted of interfacing the individual component modules with the GUI file and then proceeding with further tests using one of the data sets detailed in Experiment 1.

5.3 Experiment 1 - System Testing

The objective of this experiment was not necessarily to prove any positive influence of the techniques implemented on tweets in general. The objective was instead to affirm user acceptance testing undertaken throughout the project. The experiment was based upon a file of self-created tweets that was normalized and compared to a machine translation of what I envisaged as the 'perfect' normalization. The reasoning behind this is that it was not feasible to obtain a gold standard translation of each tweet. Since the data is primarily for testing and not evaluation the resources to obtain gold translations were allocated to Experiment 2. The perfect translation of the original tweet is simply a machine translation and since the normalized version was part of user acceptance testing there is a high level of bias. This high level of bias means that any conclusions and discussion of the results of the experiment should be taken very lightly when compared to the overall evaluation of the project.

5.3.1 User Acceptance Testing

As previously touched upon, this data set was firstly used to affirm that the system was working as intended. The testing initially began with a much smaller set of data which then evolved and grew larger as different normalization techniques were implemented. This cycle of testing was constantly supplemented by inspiration from using the much larger tweet set mentioned in Experiment 2. It was necessary to use this larger development set to reveal edge cases that would not have initially been discovered from self-creation of the tweets.

The GUI assisted greatly in the process of testing the implementation of the system. By being able to individually select which normalization techniques to apply and which tweets I wanted to test upon, I was able to refine each technique accordingly. Different combinations of techniques were applied to achieve a consistency across the system and this also allowed me to refine each technique.

5.3.2 Set Information

Source Name: N/A

File Name: test_tweets.txt

Total Size: 46

Reference Language: French (Machine Translated) Description: English tweets containing a broad variance of OOV tokens that were manually created to imitate real life tweets. This is not a true set of data. As the set is manually produced the size is small in comparison to other collections. Reference translation to be a French translation determined from a machine translation of the original tweet.

File Name	Number of Tweets	Number of OOVs
E1_T1.txt	46	126
Token Category (incorrect)	Occurrences	% of Total OOVs
Account Tags	16	0.132
Hash Tags	28	0.231
URLs	6	0.050
Fused Words	14	0.116
Excess Characters	18	0.149
Other Replacements	2	0.017
SpellChecked	35	0.264
Replaced Slang	7	0.058

Table 14: Statistics - Experiment 1

5.3.3 Results

Technique	Total Candidates	TER Score	BLEU Score
Original	0	0.533	0.794
Regex	84	0.315	0.804
Spellchecker	42	0.488	0.814
All	126	0.299	0.807

Table 15: Evaluation - Experiment 1

5.3.4 Discussion

As expected the results were very good, with the combination of techniques obtaining a 56% decrease in the TER value. However, as mentioned, this high value was expected. The broad spread seen in the token occurrences would not generally be seen in other sets of data, primarily down to the avoidance of using many pro-nouns. What was interesting in the testing was that using the spellchecker had a much smaller impact on the TER value than using the strictly regular expressions on their own.

It was interesting to see that the BLEU score had a very slight increase. The high score of all translations suggests that the the original tweet translations were very similar in meaning. It is also worth noting that simply using a spellchecker on it's own provides the best BLEU score. Again, it is worth remembering that the reference translation we compare against is not necessarily a perfect translation as any SMT system is not perfect.

Looking at both metrics, it is interesting to see that regular expression based normalization has a bigger effect on the TER score, whereas spell checking normalization affects the BLEU score more. This is likely down to the way both metrics are calculated, with BLEU being more complex and looking at combinations of words as opposed to the number of string replacements.

5.4 Experiment 2 - Development Tweets

This experiment concerns the large volume of tweets that have been used to develop the techniques in this project. By extracting two sets of data from this volume we can produce two test sets

that have been filtered out of duplicates. Normalized and original tweets will be translated into French. These two translations and additionally a 'perfect' normalization of the original tweet, will be sent to a fluent French speaker.

There will be no evaluation metric used in this experiment. Instead, the translations will be judged against each other and against the speaker's interpretation of the perfect normalization. Whichever translation is closest in meaning to the interpretation will be judged to be the 'best' translation.

As to what constitutes the better translation, this will simply be up to the jurisdiction of the speaker. The speaker will not be informed of the circumstances of the investigation nor be privy to any information regarding the project.

Two test sets will be chosen simply to break down the analysis of OOV tokens and to encourage a larger spread of token appearances. Additionally, because of the human resources required to determine the better translation the combined test set consists of only 100 tweets.

5.4.1 Set Information

Source Name: NLP Group, 2012, Sheffield University

File Name: tweets_for_analysis.en

Total Size: 143955453

Target Translation: French

Test Set 1: 50 tweets selected between lines 15000 and 20000 of the super set

Test Set 2: 50 tweets selected between lines 245000 and 250000 of the super set

Description: Plain English tweets that have been automatically sourced from Twitter using various topics and user names. There is a large amount of variance in small ranges but by randomizing tweets selected between ranges variance in tweet style can be achieved.

File Name	Number of Tweets	Number of OOVs
E2_T1.txt	50	140
Token Category	Occurrences	% of Total OOVs
Account Tags	24	0.171
Hash Tags	20	0.143
URLs	2	0.014
Fused Words	1	0.007
Excess Characters	12	0.086
Other Replacements	9	0.064
Slang Matches	3	0.021
Non-Dictionary	69	0.493

Table 16: Statistics - Experiment 2 - Test 1

File Name	Number of Tweets	Number of OOVs
E2_T2.txt	50	158
Token Category	Occurrences	% of Total OOVs
Account Tags	49	0.31
Hash Tags	2	0.013
URLs	11	0.070
Fused Words	0	0
Excess Characters	12	0.076
Other Replacements	16	0.101
Slang Matches	11	0.070
Non-Dictionary	58	0.367

Table 17: Statistics - Experiment 2 - Test Set 2

5.4.2 Results

Test Set	Equal	Original	Normalized
Set 1	25	18	8
Set 2	32	11	7
Total	57	29	15

Table 18: Evaluation - Experiment 2

5.4.3 Discussion

Set 1 was a more varied data set when compared to Set 2 with a large selection of tokens, however some tokens such as URLs and fused words were not well represented. The evaluation of the effect of normalization did not go particularly well. Out of the whole data set only 15% of of the total translations were in favor of the normalized techniques, with 16% and 14% of Set 1 and Set 2 being the percentages respectively. This is in comparison to 36% and 22% of Set 1 and Set 2, again respectively, in favor of the original tweet translations. These results would suggest that normalization has a rather small effect as a whole on making tweets more translatable. It is also worth noting that both test sets had a large amount of tweets that were considered "equal" in translation. 50% and 64% of Set 1 and Set 2 translations were considered equal which I think adds a larger margin of error to the evaluation of this experiment. The issue with with the evaluation of these "equal" translations is that for the third party, there might be very subtle differences in translation quality that might be ignored. The high percentage of equal quality translations might be explained by the short nature of tweets. If a tweet consists of very few tokens then the chances that normalization has larger than subtle effect on translation is reduced.

It can also be considered that the normalization itself is not a perfect implementation. If we consider the normalization as a process, partial normalization may in fact reduce the quality of translation. Instances of this include an incomplete supplementary corpus when looking at a specific domain of data where proper nouns are replaced by words that don't make sense as part of the sentence.

5.5 Experiment 3 - Parallel - Gold

This experiment concerns a publicly available corpus of parallel data containing high quality, gold translations. By firstly processing the data set into the JSON file designed earlier, the data was processed and normalized using the system scripts. Experiment 3 looks at the effects of individual applications of normalization techniques and the combined effect.

5.5.1 Set Information

Source Name: *utopia* Gold En-Fr Corpus

File Name: parallel.gold.en-fr

Total Size: 594

Reference Translation: French

Noise Level: Low

Description: This data set is a set of English and French tweets that have been manually collected from Twitter. There is a low level of noise, and a small amount of Twitter specific language used as can be seen by the pre-analysis below in Tabel 17. The data set is small and is not a subset of a larger data set.

File Name	Number of Tweets	Number of OOVs
E3.T1.txt	594	615
Token Category	Occurrences	% of Total OOVs
Account Tags	5	0.008
Hash Tags	15	0.024
URLs	1	0.002
Fused Words	37	0.060
Excess Characters	80	0.130
Other Replacements	5	0.008
Replaced Slang	7	0.011
SpellChecked	467	0.760

Table 19: Statistics - Experiment 3

5.5.2 Results

Technique	Total Candidates	TER Score	BLEU Score
Original	0	0.684	0.767
Regex	143	0.734	0.793
Spellchecker	474	0.738	0.794
All	615	0.738	0.794

Table 20: Evaluation - Experiment 3

5.5.3 Discussion

The results for Experiment 3 were mixed. For the TER score calculations, each process of normalization resulted in a higher TER score which does not indicate that the normalization had a better impact on the quality of translation. It is worth noting however that the number of regular expression based tokens numbered only at 24% of all tokens, indicating low noise which makes it harder to improve the translation. Even so, an 8% increase in the TER score is disappointing. It

was also surprising to see a very consistent TER score between both the single applications of normalization and the combined. I would have expected for this data set that individual application would not make much difference but for the combined normalization to improve the score.

The BLEU normalization was more positive than the TER evaluation, with a 3.4% increase in score over the whole collection for each technique. It was interesting to see that all values were very similar for the BLEU score, but that the combined score did not actually change much. Given that 76% of OOV tokens were spell checked I was surprised to see that there wasn't more disparity between the scores for singular application of normalization. Like previous data-sets the short nature of tweets is evident in that the original tweet translation scored a very high 0.767 BLEU score. This supports the pre-evaluation detailed before, that the data set is low noise and already in a good position to be translated.

5.6 Experiment 4 - Parallel - Noisy

The final experiment that was carried out on the system involved using a parallel corpus of tweets taken from the same origin as that of the data set in Experiment 3. The difference between these two data-sets is the source of collection of the tweets. Like Experiment 3, this experiment looks at the individual applications of techniques as well as the combined use.

5.6.1 Set Information

Source Name: *utopia* Gold Cn-En Corpus

File Name: data.cn-en.json

Total Size: 2021

Reference Translation: Mandarin

Noise Level: High

Description: This data set is a set of English and Mandarin tweets that have been automatically gathered from Twitter using a web crawler. As such, there is a high level of noise in the source tweets and there are instances of imperfect reference translations.

File Name	Number of Tweets	Number of OOVs
E4.T1.txt	2021	4035
Token Category	Occurrences	% of Total OOVs
Account Tags	419	0.104
Hash Tags	58	0.014
URLs	96	0.024
Fused Words	270	0.067
Excess Characters	469	0.116
Other Replacements	41	0.010
SpellChecked	2717	0.673
Replaced Slang	92	0.022

Table 21: Statistics - Experiment 4

5.6.2 Results

Technique	Total Candidates	TER Score	BLEU Score
Original	0	1.929	0.364
Regex	1226	1.847	0.358
Spellchecker	2809	1.621	0.340
All	4035	1.717	0.347

Table 22: Evaluation - Experiment 4

5.6.3 Discussion

Again, the results presented in the fourth experiment of this project are interesting and differ from results that we have previously seen. Firstly, we can see from the initial BLEU and TER values that the original tweets are extremely noisy and provide very poor translations. We can see that using the spellchecker normalization has the biggest impact on the TER score giving a 16% reduction in the score. This value is promising and given that 68.3% of OOV tokens were

spellchecking tokens. a 16% reduction is a good score. We can also see that regular expression based normalization also reduces the TER value which is a promising sign given the minority number of regular expression tokens. Combined, the TER score is reduced by 8.9% which, while still significant, is not as good as the score using a spellchecker on it's own.

For the BLEU score the results were a lot less significant. All applications of techniques failed to increase the score and instead decreased it marginally. The 'best' score achieved using techniques was using regular expression based normalization on it's own which registered a 1.7% decrease in value. An interesting facet of this experiment is that the reference translations were all in Mandarin which has a completely different set of grammar to Latin and Germanic languages and I would be interested to see if using another data set in a different language would affect the BLEU score positively.

Overall, the minor decrease in BLEU score is countered by the large decreases in TER score which does indicate that the normalization techniques have positive results. It was very interesting to see that the spellchecker decreased the TER score massively but also had the largest decrease for the BLEU score. This shows that the two scores have very different meanings in terms of evaluating the translation and is something that will be taken into consideration when judging the outcome of the project.

6 Challenges

In this section of the paper I will be discussing the main challenges that confronted the project and then discuss, briefly, potential solutions.

6.1 Slang Words

The first problem that will be discussed is that of slang words. As mentioned earlier in this paper, there exist two types of slang that are used and there was difficulty in adapting the system to normalize both of these cases.

6.1.1 In Vocabulary Slang

This type of slang involves words which are present in an English dictionary but are not common enough to be translated properly by SMT systems. In these situations the ideal normalization would be to swap out the slang word for a grammatically correct word with the same meaning. However, I was unable to produce a technique by which to accomplish this. The difficulty was researching the resources to design a method, given that we would be required to look into the 'meaning' of a tweet and developing from there. I felt that this went beyond the scope of the project.

6.1.2 Out Of Vocabulary Slang

While normalization of this area was accomplished it was only to a degree. Ideally with more resources, a complete and regularly updated dictionary could be found to normalize slang of this type.

6.2 Proper Nouns

A major problem that decreased the effectiveness of the spellchecking normalization was the issue of proper nouns. Since on Twitter the data supplied was completely randomized and taken from a wide variety of sources, the domain was also random. This meant that building a corpus of proper nouns to ignore was very difficult. The main types of proper nouns that proved to be problematic

for the project were nouns such as personal names and place names. There was no feasible way for the project to detect or add these proper nouns to the dictionary.

6.3 Punctuation

One of the features of the system was the removal of all punctuation that was not part of a token that I classified. While this made it easier to classify tokens and to perform normalization, in some cases the lack of punctuation caused alternate translations. I would be interested to see how future projects would tackle this issue. In particular, designing a system which begins to recognize correct grammar in text such as tweets would likely constitute a project of its own.

6.4 SMT Systems

While this system used Microsoft Translate as its primary source of Machine Translation there was no comparison of different systems. If the project had more financial resources at its disposal it would have been possible to have a wider evaluation by evaluating against Google Translate translations as well. It would be in the interest of this area of research to look at which SMT systems provide the best results on tweets.

An additional challenge of the system was the character limit imposed by the free license of Microsoft Translate. Limited at 20,000,000 characters a month, this limit meant that I had to be careful in development of techniques. As the project progressed, I found I had used up 85% of the character limit, meaning that further development would have been hindered had the project been continued after submission.

6.5 Reference Translations

While four different experiments were produced in order to evaluate the system, the reference translations for each data source had their own issues. While these problems can be read about in detail within the relevant experiment descriptions there are two overlying causes. The first is that it is extremely resource intensive to produce large scale gold standard translations of data set of tweets. It is far beyond the resources of this project to find a fluent speaker to set aside many hours to individually translate two separate tweets (normalized and original).

Secondly, the exact description of what an 'ideal' translation will differ from speaker to speaker. There are, for example, techniques implemented by the system that were originally adapted to completely normalize tweets and remove instances of hash tags and account tags. However, the reference translations that were acquired were all foreign language versions of tweets as opposed to a complete normalization. In future, it would be necessary to explicitly examine a large selection of tweets and to deduce and research what the optimum normalization would be with foreign language experts. Following that, the gold standard normalization's of tweets could be acquired without wasting resources on translations that were not in line with the vision of what the system should be comparing against.

6.6 Character Encoding

One problem encountered when processing the large amounts of tweets at my disposal was the issue of characters that would cause the system to crash if processed.

When using Python, it was time consuming trying to process many very obscure characters through the system. As a result of this, some of the data-sets had tweets that were causing issues with the system removed. Ideally, with more time, there would be focus put on being able to process the ever expanding character sets of the web.

7 Conclusion

As a final word in this project, I would like to discuss what I set out to accomplish and how the implementation and evaluation of the system affect that. This chapter will also talk about where the project may lead in the future and reflect on what areas of work would benefit from being developed further.

7.1 Aims & Objectives

The primary aim of this project was to be able to show and prove that implementing rule based normalization techniques over a corpus english tweet would improve the quality of translation using a machine translation system. This was to be measured using the two metrics, Translation Error Rate (TER) [14] and the BLEU metric [13], where a lower score for TER and a higher score for BLEU corresponded to measured improvement. A secondary aim relating to the primary aim was to look at how different types of normalization, namely regular expression based and spell check based, affect the normalization scores.

Additionally, the project set out to implement a full pipeline whereby sources of tweets could be extracted and normalized by scripts. Following these, the pipeline would automatically translate the tweets and evaluate the scores for TER and BLEU.

The final objective was to combine the full pipeline and evaluation using a GUI, with the option to test out single tweets by inputting a reference translation.

7.2 Successes & Failures

7.2.1 Successes

In terms of evaluating the system, the experiments carried out covered multiple data-sets all with their unique characteristics. However, none of the data sets evaluated in the experiments were perfect and each one has it's drawbacks. Despite this, the system did produce improvements in machine translating tweets in various areas. Most notably, in Experiment 3, I was able to increase the already high BLEU score on a gold standard data set. Secondly, achieving a 16% decrease in the TER score in Experiment 4 was also very good. Despite the poor performance by the system in Experiment 2, I thought that the successes in the other experiments showed that the

project was slightly successful in improving the BLEU and TER scores. With more time, I think it would be possible to tweak techniques and to be able to combine both regular expression and spellchecker normalization to work better with each other. The outcome of this being that instead of getting varied scores for each application, the combined application would produce a greater score than either single application. The analysis of the normalization techniques separate as well as combined was also beneficial to the outcome of the project. This exposed the discrepancy between the evaluation metrics and showed that the normalization of tweets depended greatly on the level of noise in data sets. The main example here being the use of the spell checker normalization in Experiments 3 and 4. In Experiment 3 we saw mixed results with a 3.3% increase in BLEU score but an 8% increase in TER score as well. Following that Experiment 4 showed a very substantial 16% reduction in TER scoring but also a 6.6% decrease in BLEU score. This combination of results, while conflicting, is beneficial for future research and shows that the project would do well in future to have broad data sets consisting of low, medium and high level of noise in the original tweets.

The full pipeline was implemented to an extent, and was very successful for the most part of the project. Using the GUI to interface various scripts and modules was possible for the project up until the point of translation. Since the scope of the project was research and not user delivery I felt it best to have the translation area of the project to be separate to the GUI. As it stands, due to the varying types of data input at the beginning of the pipeline, the system as a whole is not a complete automatic pipeline. If more time was allowed to the project I would definitely focus more effort on implementing a full pipeline and a more organized GUI.

7.2.2 Failures

While this system did have success in areas, the results were a mixed bag overall and there were a few issues that did not show normalization having a positive impact on translation quality.

The main failure of the project was Experiment 2 which did not show any positive results, with only 15% of normalized tweets containing a better translation out of the total dataset. Alternatively, original tweets had a better translation 29% of the time overall. The difference between Experiment 2 and the rest of the Experiments is the manner of evaluation, in that a human-related metric was used instead of a calculation. Arguably, this shows that the system

was not able to achieve a better 'human' judgment of quality of translation overall. On the other hand, this also potentially opens up the avenue of further research on the topic of what makes a 'perfect' normalized tweet. This would require more survey to be done on tweets in general to discover what core elements should be normalized to using a group consensus.

Secondly, the results of evaluation scores on Experiments 3 and 4 were not perfect. While, the system was able to achieve improvements in metric for either BLUE or TER in each experiment, (BLEU for Experiment 3 and TER for Experiment 4), we alternatively saw negative changes in the opposite metric. This is not ideal and therefore further research into the way these metrics work may uncover the reasons the normalization was not effective in these cases.

7.3 Further Work

Given the amount of noise present in the data set used for Experiment 4, there is a lot of room for improvement in the area of text normalization. What makes the area of tweets so interesting is that the domain and style of these tweets can be completely random and we never see any string longer than 180 characters. If I was to continue work in this project I would be very interested in research regarding common features of tweets that reach the maximum character limit of 180. I feel that this feature could lead to indications of what specific features we can expect of words present within said tweet, for example, seeing more instances of abbreviations or shortening of words. If a tweet is 179 characters long then correcting a word to a word that is longer than the original would not make sense.

Research into the punctuation and grammar of tweets would also be an interesting topic to research. If a person with significant knowledge of linguistics were to tackle the project perhaps we could see improvements in translation by leaving in punctuation when processing. This also leads down the route of how the punctuation affects the SMT system that the project works with.

A final area of research that I would think suitable for this project is that of adapting normalization of texts to be dependent on a target language. If we begin to see vast improvements in MT quality of English-French tweets, would we also see improvement in the quality of English-Mandarin tweets. While this area of research was covered very slightly, the difference between the data sets of Experiment 3 and Experiment 4 was large enough that no reasonable conclusions could be made.

8 References

- [1] - Banerjee et Al, Domain Adaptation in SMT of User-Generated Forum Content Guided by OOV Word Reduction: Normalization and/or Supplementary Data, Proceedings of the 16th EAMT Conference, 2012
- [2] - <http://norvig.com/spell-correct.html>
- [3] - Han, B., Cook, P., Baldwin, T. Lexical Normalisation of Short Text Messages. ACM Trans. Intell. Syst. 2011
- [4] - https://en.wikipedia.org/wiki/List_of_words_in_English_with_tripled_letters
- [5] - <http://www.noslang.com/>
- [6] - Balachander Krishnamurthy, Phillipa Gill, and Martin Arlitt. A few chirps about twitter. In WOSP 08: Proceedings of the first workshop on Online social networks, pages 1924, New York, NY, USA, 2008. ACM.
- [7] Max Kauffman, Kalita, Syntactic Normalisation of Twitter Messages, 2010
- [8] Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. Investigation and modeling of the structure of texting language. Int. J. Doc. Anal. Recognit., 10(3):157-174, 2007.
- [9] Phillip Koehn and Hieu Hoang. Moses: Open source toolkit for statistical machine translation. Technical report, Annual Meeting of the Association for Computational Linguistics (ACL), demonstration session, Prague, Czech Republic, June 2007.
- [10] - <http://www.alexa.com/topsites>
- [11] - <https://blog.twitter.com/2012/twitter-turns-six>
- [12] - <http://www.pewinternet.org/fact-sheets/social-networking-fact-sheet/>
- [13] - Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, BLEU: a Method for Automatic Evaluation of Machine Translation, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 311-318.
- [14] - Matthew Snover and Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul, A Study of Translation Edit Rate with Targeted Human Annotation, 2006
- [15] - https://en.wikipedia.org/wiki/Google_Translate
- [16] - <http://translationjournal.net/journal/56google.htm>
- [17] - <https://cloud.google.com/translate/v2/pricing>

- [18] - https://en.wikipedia.org/wiki/Bing_Translator
- [19] - <http://pythonhosted.org/pyenchant/>
- [20] - <https://en.wikipedia.org/wiki/Emoji>
- [21] - <http://www.cs.cmu.edu/~lingwang/microtopia/#twitter>
- [22] - Pete Norvig SpellCheck
- [23] - Tkinter Library
- [24] - http://daringfireball.net/2010/07/improved_regex_for_matching_urls