An Investigation into Noisy Language in YouTube Comments

Charles Revett
School of Computing and Communications
Lancaster University
charlierevett@gmail.com

Friday 28th March, 2014

Abstract

As the popularity of YouTube, a social video sharing platform, increases so does the quantity of comments a video creator receives from their audience. This influx in viewership has seen a need for natural language processing (NLP) tools to better extract value from comments, however the effectiveness of these tools is bounded by the high level of variance found in YouTube comments (YTC). This paper presents an in-depth linguistic examination of the noise found within YTC, partnered with a novel application that allows for unsupervised normalisation tailored specifically to the unique set of requirements present in YTC. It sets out to prove the level of variance in comments is higher than that found in other forms of short unstructured text, such as Twitter or SMS, and through the use of normalisation can improve the accuracy of NLP techniques on a corpus of YTC.

Keywords: *Natural Language Processing, Normalisation, Social Media, YouTube.*

1 Introduction

The last decade has seen the web increasingly become a host for user-generated content through the means of microblogging services and social networking sites, all labelled under the umbrella of *social media*. Combining this with the rapid expansion of Internet use, especially within young adults[38], has led to the increased importance of being able to extract value from these sources through the use of NLP[8][13]. However techniques such as Sentiment Analysis[16] and Information Extraction[56] often perform poorly in this domain due to the unstandardised casual style of writing characterised by social media users. NLP tools such as the Stanford Named-Entity Recognizer saw performance drop from 90.8% to 45.8% when used on a corpus of Twitter messages[41]. This confirms the need for pre-processing capable of cleaning noise from social media text, thus allowing existing previously successful NLP techniques to work accurately.

The approach proposed in this paper focuses on crafting a novel application capable of efficient unsupervised normalisation of variance found in social media text. The proposed system will utilise a model constructed from automatic tokenisation, combined word replacement scheme, a pre-defined social media word list and context aware correction through the use of common N-Grams. Each component will rely heavily upon a handcrafted

Working Documents: `http://lancs.ac.uk/ug/revett/320`

dictionary which automatically evolves to include language found within new social content on the Internet, and was also influenced by the manual normalisation of 2,000 YouTube comments. The application has three primary purposes:

1. Pre-processing for NLP tasks, specifically those which are automated and are oriented around the use of a corpus of noisy text, especially if it originated from a social media site.
2. Normalising text in 3rd-party applications, thus helping non-native users to understand the irregularity of text and removing the barrier to fully participate.
3. To become part of a much larger NLP system used by YouTube video creators looking to extract value from their audience's comments through techniques such as; automatic summarisation, sentiment analysis and question extraction.

1.1 Motivation for YouTube Comments

YouTube states that it currently processes 100 hours of new video submissions a minute and has over 1 billion unique users a month, that combined watch over 6 billion hours of video each month[2]. These statistics justify the first reason YTC were chosen for the corpus of this paper, as it is clear that YouTube is a rapidly expanding powerhouse[17] within the field of social media.

As YouTube's user base continues to increase so does the quantity of comments video creators are seeing their audience post. This poses a new problem as creators are no longer able to keep up with the stream of comments and thus are losing the value they contain. Through the use of NLP, creators could see the value from the comments be extracted automatically. Concepts applicable for use with a corpus of YTC could be; sentiment analysis (opinion mining), question extraction (common audience questions) and automatic summarisation (reducing the amount of text to read). However this is not currently possible due to the high amount of linguistic variance in YTC.

This paper defines noise in text as any kind of variance which does not conform to the expected rules of spelling, grammar and punctuation[60]. All three types of variance can be found in YTC, however the overall written quality of a comment varies significantly as well as the patterns of variance seen within comments. Section 3 of the paper outlines a ranked list of the most common variance patterns found in YTC, compared across different genres of videos as well as compared to other forms of short unstructured text (Twitter and SMS). This justifies the second reason why YTC were chosen, as the level of variance found within comments (described in Section 3) is greater than that of any other social media text. Alongside this, there has been no formal analysis of the language used within YTC published previously.

The motivation for this paper is reinforced by an interview carried out with the popular YouTube video creator *Pyropuncher*[1]. Since joining YouTube in May 2007 he has published over 1,750 videos and amassed over 69 million views from his audience. He confirmed the continual growth YouTube has received, the difficulty he has keeping up with the stream of comments from his audience and the importance the commenting system serves:

> *"Over time as my fanbase increased the views and comments increased with it. [...]Comments are vital to see what type of content the audience wants. [...]Nowadays it's hard to read every comment with most videos getting hundreds of comments."*

He then went on to confirm the presence of variance within YouTube comments and the need for an automatic solution:

> *"People treat YouTube as a casual forum so there is a lot of spelling errors on it, especially from younger members. [...]A general summarised message from the comments would be great and have a lot of value."*

1.2 Characteristics of YouTube Comments

Normalisation models have traditionally targeted general spelling variation within written text, relying heavily on phonetic similarity of words to offer corrections. This method performs poorly however when processing a corpus of social media text due to the candidate correction word often lying outside the specified edit distance. Alongside this challenge, social media text presents two further difficulties.

The first being the vast number and diversity of non-standard tokens. Using the Edinburgh Twitter corpus[53] as an example, Liu et al. quantified that of the 97 million tweets there were over 4 million unique unrecognised tokens[40]. The second challenge is the reasoning behind an unidentifiable token being written. Many users may unintentionally include variance in a message through misspelling for example, however other users may purposefully add variance for a number of reasons, such as; speed, slang, sentiment, emphasis or ease of typing[60]. Furthermore, the effortless simplicity of being able to instantly post a message to a social media site, without the presence of an editor, makes it all the easier for users to add variance. This method of submitting text into the public domain had never been seen before the advent of social media and is vastly different to the process of writing and editing a newspaper article, for example.

Surveying YouTube comments specifically, they can be seen as a method for independently broadcasting a piece of information and/or as a method of asynchronous communication between multiple collaborators. There is no defined set of reasons for why a user will comment, and each comment, although being limited in scope to a particular video, may not actually target its meaning at the video itself. This in turn highlights the very erratic nature of YTC where submissions are often unusable for the video creator[59], thus offering one insight into why the variance is so high.

1.3 Problem Statement

The primary objective of this paper is to investigate if YTC can be accurately normalised without supervision. This paper will also provide a linguistic analysis of a corpus of YTC, and the categorisation of the different types of variance found within it.

The paper presents two separate quantitative based evaluations of the results gathered. The first being an evaluation of the accuracy at which the system corrects errors, accompanied by an examination of changes seen in the BLEU (Bilingual Evaluation Understudy) metric thus highlighting any alteration in the quality of the text subsequent to normalisation. Secondly the size and quality of the dictionary will be evaluated. This will be carried out through reviewing how accurately specifically sized dictionaries detect errors, quantified by an assessment of the precision and recall. It is important to note that each evaluation will utilise a completely independent 2,000 word corpus of YTC, that had no interaction with the system prior to evaluation. Each evaluation will address the following hypotheses respectively:

> *A hybrid approach to normalising text is capable of automatically correcting variance found in YouTube comments to a high standard without supervision.*
>
> *Careful attention to the overall size of the dictionary used by a normaliser, and the methods utilised to populate it, increases the overall accuracy of a given system.*

The contributions of this research can be summarised as follows:

1. A novel unsupervised normalisation system is introduced, aimed specifically at cleaning noise from YTC.
2. A novel investigation of the language and variance observed within YTC is presented.
3. A structured corpus of YTC is made available for further research by interested parties (upon request).
4. Two high quality dynamic linguistic datasets are made available to assist interested parties in future research.

The remainder of this paper is arranged as follows. Section 2 provides an understanding of the background to text normalisation, paying close attention to previous unsupervised systems, systems aimed at normalising social media text and each of the components within the proposed solution. Section 3 provides an examination of the types of language and variance seen within YTC as part of a linguistic analysis, with comparisons to other social text (Twitter and SMS). Section 4 describes how the proposed normaliser attempting to correct variance in YTC was implemented, including the NLP concepts featured in the system. Section 5 offers two independent quantitative based evaluations of the proposed application. Section 6 concludes the paper by focusing on how well the primary objectives have been addressed, any limitations to the proposed solution, the implications of the results gathered and any future work that could be done to extend the findings of this research.

2 Related Work

This section gives an overview of the previous research that this paper uses as its foundation, looking specifically at traditional methods for correcting spelling errors and previous systems attempting to normalise social media text. Alongside these two main categories of background research, the paper also examines previous work relating to components of the proposed solution.

2.1 Traditional Spell Checkers

Work began as early as the 1960s[21] on the implementation of methods for detecting and correcting spelling errors within standard text. Since then there have been two comprehensive reviews of the state of the literature relating to spelling correction, provided by Peterson (1980)[51] and Kukich (1992)[37]. Each review groups research into three main categories; the detection of unknown strings within an body of text (non-word error detection), the correction of non-word spelling errors in isolated words and the correction of real-word spelling errors in context-dependant words. This section examines each independently, as well as inspecting the variety of ways in which systems have structured dictionaries in the past.

Kukich states that the distinction between spelling error *detection* and *correction* is an important one to make. This is demonstrated by the fact that the task of detecting words that do not appear in a given lexicon already has multiple efficient solutions; however the task of ranking a list of candidate words and selecting one with which to correct the spelling error is a never-ending pursuit. Bentley[10] illustrates this problem by pointing out that the rate at which words enter and exit the language makes spelling error correction very difficult. This problem of having an outdated dictionary is linked very closely with social media text due to the informal nature of language used and how it constantly changes to reflect emerging social trends.

2.1.1 Non-word Error Detection

This area concerns the detection of strings which do not appear within a given predefined dictionary. A significant amount of work was done from 1970 to 1985 targeting the development of comparison techniques and pattern-matching algorithms capable of highlighting unknown strings within a body of text.

The most basic example of this is the UNIX *Spell* system[43] which formed a list of strings it believed to be misspelled by highlighting any string which was not found within a pre-defined 25,000 word dictionary. This straightforward method highlighted two issues; the use of a static predefined dictionary limits the accuracy of the system in the future, and large dictionaries heavily impact on the speed of the operation.

Furthermore, Peterson found that larger dictionaries result in a much higher number of undetected typing errors[52]. His research found that as high as one in every six typing errors could go undetected when using a large dictionary. Walker and Amsler[66] strengthened Peterson's argument for a shorter word list, as it was found that the words in the *Merriam-Webster Seventh Collegiate Dictionary* and those in the eight million word *New York Times* news wire corpus were significantly mismatched when compared. 61% of the words observed in the dictionary were not found in the corpus, and conversely 64% of the words in the corpus were not found in the

dictionary. This clearly highlights the importance of keeping the dictionary to a minimal size and scoping the language specifically to a single field (e.g. medical).

The performance impact of large dictionaries was solved through methods which reconstructed the dictionary, each have been reviewed by Turba[64]. The most common solution is to populate a hash table with addresses of known words using a hashing function[43][47]. To check a given string, its address is computed using the same function and validated using the hash table. This method only requires a single validation, and relies upon the need for a fast hashing algorithm that takes up minimal disk space. The early 1990s saw an array of research targeting fast hashing of variable length text strings, most notably by Pearson[49] and Fox et al[27].

Alternatively a dictionary could be reconstructed as a series of length-segmented lists. The input string would be checked against all known words of the same length, thus reducing the size of word list used for validation. The length of the word list used can be reduced further through checking the input string against all known words of the same length and starting letter[64].

The process of *affix-stripping* can be combined with the other techniques discussed to further reduce the size of a word list. Space can be decreased by implementing a set of rules which remove the prefix/suffix of a word before validating it through the use of a dictionary. McIlroy[43] used this technique to reduce the size of the UNIX *Spell* system's word list by a third.

On the other hand a system can operate without the need for a dictionary through populating a list of n-grams sorted by frequency. Cavnar describes a *n-gram* as a n-character slice of a longer string[15]. Riseman and Hanson[55] implemented the use of n-grams by comparing each trigram within a string to a pre-populated table of trigrams, and highlighting any string as an error if it contained an unknown trigram. Morris and Cherry[46] later expanded on this work by instead dynamically populating the frequency table of trigrams from the given body of text. The use of n-grams is limited in scope however as a high proportion of common spelling errors do not contain unknown trigrams, thus resulting in this method's value being limited to optical character recognition (OCR)[32].

2.1.2 Isolated Non-Word Error Correction

This area concerns the correction of a single word detected as a spelling error, without taking into consideration the context that it sits within. The process is split into two tasks; generating candidate correction words, and ranking candidate correction words. In the following discussion, previous techniques[1] are described in terms of both of the subtasks, looking specifically at:

1. Similarity-based techniques.
2. Rule-based techniques.
3. Minimum edit distance (MED) techniques.

Similarity-based techniques simply compute a representation key for all given strings, such that the comparison of similarly spelled strings results in identical or similar keys. The *Soundex* system is the best example of this technique, first patented by Russell and Odell[57] in 1918 and later implemented into the context of spelling correction by Davidson[23] in 1962. The system populates a list of words from the dictionary which have the same Soundex key as the given misspelt string - for example, '*disapont*' and '*disappoint*' share the key D215.

Pollack and Zamora devised a technique called SPEEDCOP which built upon these foundations to provide single-error corrections to misspellings in scientific text[54]. This was done by iterating through a key ordered dictionary searching for the first single-error variation of the word, where the starting point was the dictionary index of the misspelt string's key. Tenczar and Golden formulated a very similar technique (95% accurate) which differed due to a far more complex key computation algorithm which took into account variables such as word length and syllabic pronunciation[63]. More recent research has seen the need for a pre-populated dictionary

[1] Kukich[37] compares each technique's accuracy on a 170-word test set using varying sized dictionaries (p.410).

of keys vanish, instead a similarity measure can be computed for a given misspelt string and a known word by measuring the longest matching substrings in both strings[11].

Rule-based techniques attempt to represent spelling error patterns in a set of rules which quickly correct the most common misspellings within a given text. A manual analysis of a given corpus highlights the most common errors (as described in Section 3). Yannakoudakis and Fawthrop created a set of 1,377 rules which yielded a correction accuracy of 68% when tested on 1,534 misspelt strings[67]. The system generated a list of candidate correction words by searching a length-segmented dictionary for single-error or double-error variations of the misspelt word which abide by one or more of the rules.

Minimum edit distance techniques compute the smallest cost sequence of 'edit operations', also referred to as the *Damerau-Levenshtein metric*, needed to change one string to another, as outlined by Wagner[65]. Damerau[22] and Levenshtein[39] pioneered this type of spelling correction algorithm in the 1960s, where each dictionary entry was compared against the misspelt string to compute its MED. This significantly inefficient method for ranking possible candidate correction words was improved upon by both Gorin[28] and Durham et al.[26] by implementing the ***reverse*** *minimum edit distance* technique. This approach populates a list of candidate correction words by generating every possible single-error variation of the misspelt string and then removing any which are not found in the dictionary, thus processing far fewer comparisons but limiting corrections to single-errors.

2.1.3 Real-Word Error Correction

This section discusses techniques that deal with the correction of spelling errors within context-dependant words, also referred to as ***real-word*** errors. These are described by Mitton[44] as correctly spelt words needing to be substituted for another due to not being the intended word and in turn distorting the clarity of the sentence (e.g. ***b****oat* $\rightarrow$ ***c****oat*). As discussed in Section 3.3.1, the percentage of real-word errors varies significantly based on the corpus of text in question (40% in the case of Mitton's paper). There are two main areas of techniques utilising the context surrounding a word; traditional NLP and statistical language modelling (SLM).

NLP prototypes dealing with malformed text generally focus on a restricted set of tasks within a specific domain, and thus are not well suited for normalising *unrestricted* text. A typical system is built around a parsing model, supported by a dictionary and set of structured grammatical rules (grammar). Systems implementing a *relaxation-based technique* achieved the best results when processing unrestricted text and therefore will be the focus of this discussion. The EPISTLE critique[33] writing aid corrects non-word errors before parsing the text for common specific real-word errors which are syntactically incorrect. These are located by continually parsing the same string with a changing unique set of grammatical rules until the system reaches a successful parse iteration. The rule(s) missing from the successful set of grammatical rules highlight to the system the syntactically incorrect parts of the string. A Dutch text-editor[35] executed a similar structure supported by a larger grammar of 500 augmented phrase structure rules, where it instead replaced words which violated part of the grammar with a candidate word from a set of orthographically and phonetically similar words.

SLM techniques for correcting errors use *n-grams* to estimate the conditional probability that a word will occur within the context of others. An effective SLM relies upon a set of probability estimates derived from a corpus sized at 10 million words or more[37]. Through utilising contextual information a probability can be computed, where low-probability denotes a real-word error and high probability denotes a possible correction candidate. Church and Gale[18] made use of word bi-gram probabilities to enhance the accuracy of isolation non-word spelling correction techniques (as outlined in Section 2.1.2) from 87% to almost 90%. A similar use of tri-grams by Mays et al.[42] saw a Bayesian probability score computed for each sentence from the sum of each word's trigram probability multiplied by the probability the word is not a real-word error. This probability score could then be used to examine if some other sentence had a higher likelihood of being correct.

Viewing the previous literature to date in the domain of *Spell Checking*, it can be noted with certainty that the field has lacked any substantial effort to explore the accuracy of each technique discussed above on other forms of domain specific or computer-mediated communication (CMC) text. The restricted nature of the previous research achieved so far has not only to the most part ignored text originating from CMC but also subjects such

as historical text, until the work of Baron and Rayson[7]. Due to this major limitation, the concepts and their respective results discussed previously can be taken as an indication for how to implement the proposed solution but in reality the system must be built around a series of testing and evaluation iterations.

2.2 Social Media Normalisers

YouTube comments have had little attention from the scientific community, however two studies bolster the motivation for this paper; Ammari et al.[3] investigated a method for filtering meaningless comments, and Schultes et al.[58] examined the relationship between the genre of a video and the number of comments it receives. Apart from this limited collection of papers the majority of work around normalising social text has been completed using a corpus of Twitter messages, SMS texts, chat room messages or emails. This section discusses a number of much more recent systems, aimed at correcting variance found in Twitter and SMS messages.

Han and Baldwin[30] implemented a system that generates a selection of candidate correction words based on morphological and phonetic similarities to the error string without the need for annotated training data, similar to Church and Gale's SLM implementation. They achieved an accuracy of 71.2% when tested on a corpus of Twitter messages, and noted that their system's error detection approach could be radically improved as it was limited by the noisiness of the corpus. Kaufmann and Kalita[34] took a different approach and instead simplified the Twitter corpus prior to executing well-known spelling correction techniques, through the use of syntactic (rather than lexical) normalisation to remove noise. With the intent to act as a pre-processor, the system saw an average increase of 18% in the quality of text (0.7985 BLEU score) after normalisation had taken place. Clark and Araki[19] shared a similar goal of implementing a pre-processor (CECS) for normalising *casual English* aimed at being utilised by open-source spellcheckers. The paper lacked any detail on the correction algorithm used, however it did note that the authors used a pre-compiled 1 to 1 word replacement dictionary. It should be noted that this vastly increased the speed as the system is simply performing a dictionary lookup. The efficacy of two open-source spellcheckers were examined after pre-normalisation performed by CECS had taken place, using a corpus of 100 Twitter messages. The results showed the average percentage of errors in a sentence drop from 15% to below 5%, and thus are surprisingly positive due to limited correction model implemented. The most recent system to attempt successful Twitter normalisation was TwitIE; an open-source information extraction pipeline for microblog text, concentrating specifically on Twitter[12]. When evaluating the system's named entity recognition (NER) performance against a corpus of Twitter messages, it achieved almost double the F-measure of the Stanford Named-Entity Recognizer (41% compared to 80%). The normalisation module within the system shares two components to the system proposed in this paper; pre-defined social media word lists and a combined word replacement scheme - thus highlighting the success of a well-devised hybrid normaliser.

Moving to the discussion of SMS normalisation systems allows for a far greater amount of previous literature to be examined. Kobus et al.[36] proposed a novel hybrid system by combining techniques taken from machine translation and speech recognition. They concluded that the system was restricted to the purposes of social text mining as it exhibited an average word error rate (WER) of 11% and a 60% chance that an error will occur at least once in a single message. This type of hybrid system was improved upon by Beaufort et al.[9] who implemented a similar system capable of a 9.3% WER and a maximum BLEU score of 0.83. The improvement was put down to the creation of models only learnt by using a training corpus and applying different models based on whether a given sequence of tokens is known. However the authors did note that greater detail was necessary when dealing with phonetic similarities found in SMS messages. Cook and Stevenson[20] analysed a corpus of SMS messages to derive a set of common variance patterns. Utilising this set, the system was capable of suggesting the necessary correction for a given error 59% of the time. Aw et al.[4] instead approached the problem of normalising SMS messages by viewing them as though they are a different language and thus need translating to English. Making use of a phrase-based statistical machine translation model, the authors achieved a BLEU score of 0.80702 compared against a baseline of 0.6958 when evaluated on a corpus of 5,000 sentences extracted from SMS messages.

Examining the previous attempts at normalising *social* text, it can be clearly noted that there continues to be

a wide variety of techniques proposed as solutions, with no definitive answer to the problem. However with this in mind, many of the more robust techniques rely upon a hybrid model of combined techniques proven to yield successful partial results in the past. This trend has been taken into account when selecting the components to use for the foundation of the system proposed in this paper. It should also be noted that the majority of the systems discussed above feature the same limitation; they each went about attempting to normalise a given type of social text without carrying out some form of linguistic study. Thus by relying on a previous analysis of the language featured in the given text, the authors limited themselves to scoping the system to known trends without attempting to discover new ones which would have aided the development of far more successful system.

Many of the studies discussed in this section utilise the BLEU metric to evaluate the quality of text which has been translated from one natural language to another by a given system[48]. However it should be noted that both this metric and the very similar NIST metric have a number of limitations[68]. Arguments have been put forward which detail that BLEU lacks the ability to deal with languages without word boundaries[24], and that an increase in a BLEU score does not guarantee an improved translation as illustrated when human evaluations failed to match scores generated by BLEU[14]. Taking the limitations of the metric into consideration, the paper will still make use of the BLEU metric as it remains the standard for evaluating the performance of machine translation systems.

2.3 Components of Proposed System

This section highlights recent systems employing similar components as those described in the proposed system of this paper; specifically examining pre-defined social media specific word lists and previous attempts at creating a dynamic dictionary based on text extracted from the web.

The TwitIE system[12] took advantage of two sources of pre-defined social media specific words. The first contained high variance words such as "2moro" (*tomorrow*) and "brb" (*be right back*), similar to the dictionary created specifically for microblogs by Han et al[31]. The second being a list of unambiguous named entities often only found with the domain of social media, such as "PewDiePie" (famous YouTube video creator), similar to how Derczynski et al.[25] implemented a part-of-speech tagging system specific to Twitter. Thus through the use of pre-defined word lists the number of common errors made by adapted normalisers is reduced, as the need to recognise high variance or unknown words is limited.

Both Han et al.[31] and Gouws et al.[29] successfully implemented methods for extracting domain specific lexical variants of known words from Twitter. This shows that Twitter, and in turn other social media sites, can act as a valuable reference for bolstering the reliability of a dictionary in the context of normalisation. However, no previous system has aimed to utilise a dictionary which continually evolves by parsing new content on the web for new words. Previous systems have instead relied heavily upon a carefully constructed static dictionary that adheres to the target domain of the paper.

3 Linguistic Analysis

This section presents a novel examination of the variation found in YTC. Starting by introducing the purpose and reasoning for including a linguistic analysis, it then moves on to describe the different corpora used to act as a comparison to the results found. Finally it concludes by reporting on the use of two existing tools for manual normalisation; VARD2[6] and DICER (Discovery and Investigation of Character Edit Rules)[7], and the examination of any of variance patterns found. This component of the paper follows the same structure as previous papers[62] where linguistic analysis was the primary goal.

The aim of this section is to prove that the variance observed in YTC is higher than that of previously examined social texts (Twitter, SMS). This will be done by systematically categorising the most common variance patterns in YTC, and comparing any results found to the same analysis conducted on corpora of Twitter messages and SMS messages. Alongside this fine grained manual analysis, high level comparisons will also be made such as

the overall level of variance and real-word error rate. By conducting this study it will emphasize the motivation behind why YTC were chosen, and through the study of the variance trends discovered the accuracy of the system proposed in this paper may also improve.

3.1 Sources of Data

This study makes use of three corpora of different types of social text; YouTube comments, Twitter messages and SMS messages, all of which are described in a Tables 1-3 below. All bar the last corpus have been collected for the purpose of this linguistic analysis. The SMS corpus, CorTxt, was collected for the purposes of a linguistic analysis conducted by Tagg[61] and is mainly populated of text messages from an educated and literate network of the author's friends and family, therefore adding a limiting factor to the results of the study. It should be noted that any description of the corpus has been taken from Tagg's linguistic analysis, and all comparisons made to the variance found in the CorTxt paper is using the results of a separate linguistic analysis of CorTxt which employed the same techniques as seen in this study[62]. Thus Table 3 details the proportion of the CorTxt corpus which was randomly selected by Baron and Rayson[62] to manually normalise as part of their study, however that equates to only a fifth of the whole corpus.

Table 1: YouTube Comments Corpus Description

No of Comments	2,000
No of Words	24,013
Average No of Words per Comment	12.01
Collection Period	January 2014
Collection Method	YouTube Data API V2[2]
Language	English

Table 2: Twitter Messages Corpus Description

No of Messages	5,000
No of Words	45,459
Average No of Words per Message	9.09
Collection Period	March 2014
Collection Method	Twitter Public Streaming API[3]
Language	English

Table 3: CorTxt Description

No of Messages	2,430
No of Words	41,342
Average No of Words per Message	17.01
Collection Period	March 2004 - May 2007
Collection Method	Friends and Family (41% M / 59% F)
Language	English (mainly British English)

[2]YouTube Data API V2: `https://developers.google.com/youtube/2.0/developers_guide_protocol_comments`
[3]Ruby client for streaming public Twitter messages: `http://rdoc.info/gems/twitter`

The corpus of YTC collected for this study was deliberately split into 4 equal parts, each comprising 500 comments extracted from videos of a particular genre; *comedy*, *education*, *gaming*, *vlogs*[4]. This was done so that it would be possible to compare the differences within the YouTube social platform, as well as comparing the entire platform to others. It should also be noted that both the Twitter and YouTube corpora are limited as both the Twitter and YouTube APIs did not allow for detailed specifications to be chosen for the selection of comments/messages, thus resulting in no control over the audience composition.

3.2 Manual Normalisation

This study required the manual normalisation of 2,000 YTC through the use of VARD[6]; a tool used to insert modern equivalents in the place of variants to allow for comparisons to be made between the original and modern string. VARD was originally designed to manually normalise variants found in Early Modern English. However since its creation it has been adapted to deal with other sources of variance, most notably SMS messages[62] and second language learner data. As a researcher manually normalises variants highlighted in a given text, VARD will append XML tags to the document which can later be used for analysis by other tools, for example:

```
<normalised orig="u">you</normalised>
```

The above XML tag format is accepted by the second tool utilised in this study; DICER. By analysing the difference in characters between the original and modern string DICER is able to generate a list of variance rules or *trends*. Each trend can be ranked by a number of characteristics, such as frequency or difficulty for automatic normalisation (n character edit, where the difficulty increases as n increases). Unlike similar studies of this kind, a single researcher was tasked with manually normalising the randomly selected corpus of YTC. As described by Baron and Rayson[62], it is far more beneficial to have multiple researchers repeating the process of normalising text by hand to secure a far more robust set of results, thus highlighting a limitation to the results of this study.

3.3 Linguistic Comparison

The manual normalisation procedure conducted for the purpose of this study highlighted a total of 2,348 variant normalisations, therefore determining that 9.87% of the sample 2,000 YouTube comment corpus can be considered to be variant. As shown in Table 4, this is the highest level of variance discovered to date in social text.

Table 4: Corpus Variation Comparison

Corpus	**Variants**	**Tokens**	**Variance**
YouTube Comments	2,348	24,013	9.87%
Twitter Messages	3,632	45,459	7.99%
SMS Messages	3,166	41,342	7.66%

The impact of these findings clearly stresses the significance of YTC as a linguistic genre as a whole. Be that as it may, when exploring the results acquired from the manual normalisation at a finer level by investigating the variation found in YTC across genres, the level of variance fluctuates from a low of 5.67% (*education*) to a high of 15.67% (*vlogs*), as illustrated in Table 5.

Table 5 clearly shows that the length of a comment that a viewer makes on an education video is on average twice as long as those made on any other genre. For this reason, 43% of the tokens in the corpus of 2,000 randomly selected YTC originate from educational videos which in turn also have the lowest level of variance. Taking this into account, a variance level of **11.19**% would be a far better reflection of the true level of variation as it was calculated with an even weighting assigned to each genre's total tokens.

[4]Vlogging has exactly the same intent as blogging, however through the medium of video instead of text.

Table 5: YouTube Genre Variation Comparison

Corpus	Avg. Words per Comment	Variants	Tokens	Variance
Vlogs	8.05	629	4,025	15.67%
Gaming	10.34	612	5,170	11.84%
Comedy	8.99	522	4,496	11.61%
Education	20.64	585	10,322	5.67%

The DICER analysis also stressed how difficult automatic normalisation is in the case of YTC, as only 53% of variants could be corrected by a single character edit. This however is a much higher proportion than that found in SMS messages (28.5%) thus raising the argument that SMS messages may have a lower rate of variance but the variance itself is more complex and thus more problematic to normalise automatically. The figure of 53% is far closer to that displayed by more formal written texts examined by Mitton[44]. He observed that 69% of variants could be solved by a single character edit, thus prompting to question if the variance in YouTube is in fact the most difficult to normalise automatically or just that it occurs more commonly.

The question entertained above, that SMS messages have a lower rate of variance when compared to YouTube comments but the variance itself is more complicated, is bolstered when weighing the differences in the proportion of insertion, deletion and substitution operations required to normalise each of the respective texts. Baron and Rayson found that the vast majority of SMS normalisations required the insertion of new characters (72%), with only 27% requiring substitutions and less than 1% deletions. The identical ranking of edit operations is reflected in YTC, however insertions dominate to a lesser extent (52%) and deletions to a much larger extent (12%). The levels observed in YTC are again closer to those from a study of more formal text by Mitton[45] where 33% of normalisations required insertion of new characters, consequently showing that the variance discovered in YTC shares similarities with more formal written text and therefore could be considered to be of a more standard nature.

Secondly the comparison of the operation proportions in SMS messages and YTC brings to attention two further points. Foremost they highlight that in both corpora users most commonly reduce the overall length of a word for the benefit of typing faster (e.g. `'you'` → `'u'`). Secondly, it emphasises the fact that YouTube users commonly utilise the addition of characters to emphasise the importance of a given word (e.g. `'soooooooo cooooool'` → `'so cool'`). Compare this to SMS messages where the deletion of characters is under 1% of the operations conducted, therefore highlighting that the addition of characters for emphasis is linked to the user typing on a full-sized keyboard instead of a mobile device.

3.3.1 Real-Word Error Rate

The manual normalisation also allowed for the calculation of the real-word error rate, as discussed in detail in Section 2.1.3. Of the tokens found to be variants only 4.92% could be considered to be real-word errors. When compared to SMS messages (10.17%) this figure is over half of that observed by Baron and Rayson[62], and far lower than that observed by Pedler and Mitton[50] when analysing dyslexic writers' spelling errors (31.4%). The figure is very similar, although slightly higher, than the rate (2.75%) observed in Early Modern English (EModE) by Baron[5]. As discussed in Section 4.2, this is an important discovery as it reduces the need for the system to employ complex contextual algorithms for detecting real-word errors.

3.4 YouTube Comments Variance Trends

Moving on to specifically examine the most common variance found in YTC and the rules that can be extracted from them, it is immediately apparent from inspecting Table 6 and Table 7 that all of the most common normalisations feature the insertion of new characters into short simplified words. The rules found by DICER, as

stated in Table 7, largely reflect the most common variant normalisations found in Table 6 and both tables are extremely similar to those found in Baron and Rayson's study of SMS messages.

Table 6: Most Common Variant Normalisations

Variant	**Normalisation**	**Frequency**
u	you	80 (6.68%)
im	i'm	25 (2.09%)
dont	don't	20 (1.67%)
da	the	13 (1.09%)
wat	what	12 (1.00%)
vid	video	10 (0.83%)
r	are	10 (0.83%)
ur	your	10 (0.83%)
thats	that's	9 (0.75%)
hes	he's	9 (0.75%)

Table 7: Top Ten Most Frequent Normalisation Rules

Rule	**Frequency**	**Top Position**	**Examples**
Insertion '	117 (7.91%)	Penultimate (63.25%)	dont, youre, its
Insertion yo	92 (6.22%)	Start (100%)	u, ur
Insertion e	73 (4.94%)	End (50.68%)	r, hav
Insertion space	66 (4.46%)	Middle (74.24%)	sucha, alot, thankyou
Insertion a	55 (3.72%)	Middle (38.18%)	r, wht, tht
Insertion u	35 (2.37%)	Middle (85.71%)	fny, gy
Insertion o	33 (2.23%)	Middle (66.67%)	dnt, yu, tmrw, nt
Insertion g	27 (1.83%)	End (100%)	goin, comin, amazin
Insertion h	24 (1.62%)	Second (62.50%)	wat, yea
Insertion eo	20 (1.35%)	End (50%)	vid, vids, ppl

All bar the fourth most common normalisation of `'da'` → `'the'` are covered by the ten most frequent normalisation rules. Therefore because of this observation, it can be stated with confidence that the normalisation rules are mostly determined by the prevalence of specific variants (e.g. `'u'`) instead of a common trend running throughout multiple variants, which is in line with the analysis of SMS messages. This is an important point to make as it emphasises the importance of populating a pre-defined word correction list to handle the most common variants in the proposed solution. The analysis carried out by DICER not only points out the trends within YTC but also allows for future study to be carried out to further categorise each trend. As outlined by Baron and Rayson[62], these categories would help to normalise the most frequent spelling variants both manually and automatically, thus highlighting future work for this research topic.

4 Proposed Solution

This section presents how the proposed application was designed and implemented, accompanied by an examination of the normalisation techniques at its core as well as the reasoning for why each was selected.

The system partitions the process of normalising a given string into four high-level sub-processes; automatic tokenisation, error detection, error correction and correction substitution. Figure 1 depicts the overall end-to-end flow of data through the application and the individual components involved. Each column in the figure describes the input/output (white boxes) and actions (black boxes) taken by one of the high-level sub-processes. This is then followed by an in-depth description of each sub-process and the design decisions that dictated how they were implemented (Section 4.1 to 4.4).

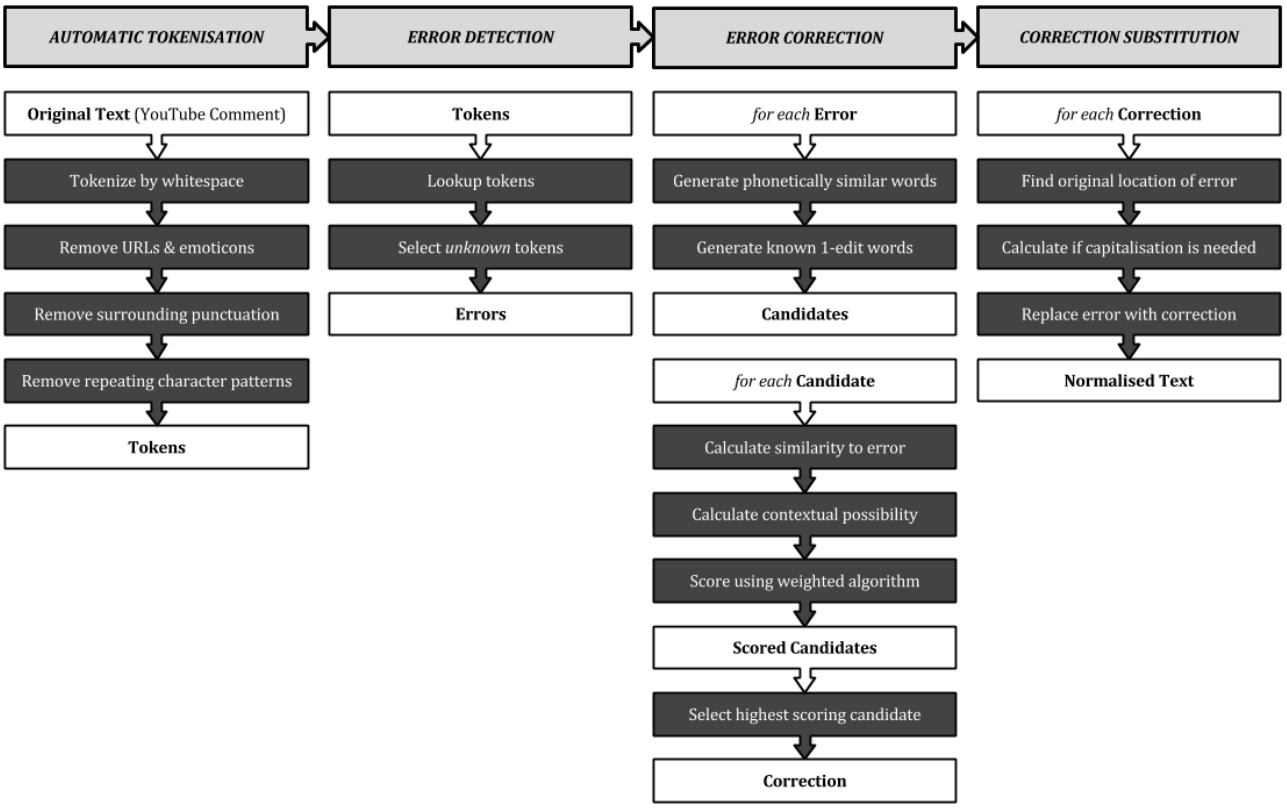

Figure 1: Diagram illustrating the end-to-end flow of data through the application. Each column describes one of four high-level sub-processes.

4.1 Automatic Tokenisation

When dealing with social media text there are a number of added token types that must be dealt with, thus general purpose tokenisers that rely on fully regulated text formats have to be adapted in order for them to accurately extract tokens from strings. Examples of addition token types found in social media text are; hashtags (ubiquitous across all social media platforms), emoticons and URLs.

With this in mind, a new tokeniser was implemented with a simple design pattern; *to extract as many tokens as possible from a given string, to then remove or alter tokens based on a pre-defined set of rules specific to the requirements of tokenising YouTube comments.* This grants the tokeniser the added benefit of being flexible to the ever changing linguistic trends found in YTC as new rules can be added if necessary.

The following set of rules were mainly dictated by the results gathered from the linguistic analysis detailed in Section 3 and comments from the researcher who carried out the manual normalisation of 2,000 YTC, however the set was adapted throughout development when necessary:

Remove: *removal of parts of a given token*

Repeating characters e.g. `coooooooooool` → `cool`

Repeating patterns e.g. `hahaahhhaahahahah` → `haha`

Surrounding punctuation e.g. `"Hello?".` → `Hello`

Ignore: *ignoring specific tokens*

Emoticons e.g. `:)`, `B-)`, `XD`, `<3`

URLs e.g. `http://google.com`, `google.com`, `www.google.com`

Numbers e.g. `143`

By ignoring a subset of the tokens that are not considered to be words, the system limits the end-set of tokens to words that the application is capable of correcting and in-turn reduces computational load. Furthermore the need for removing parts of a given token is also a decision to reduce computational load but additionally to ensure the *purity* of a token. Starting by looking at the former point, the removal of repeated character patterns from a string reduces the need to generate candidate correction words in some instances, for example `'cooooooool'` is reduced to `'cool'` which is a known word and thus no longer needs to be corrected. The latter point is instead targeting the maximum accuracy capacity of the application, as by removing any surrounding punctuation the system is making sure that only the word is left to validate.

The tokeniser also takes into account if a given token is capitalised or composed of all upper-case letters. This aids the system when having to substitute corrections back into the original string as case sensitivity is conserved, and in-turn helps to maintain any readability the string already had.

4.2 Error Detection

As detailed in Section 3.3.1, YTC have a far lower real-word error rate than other more formal text and social media text. Thus taking this into consideration, the application will only highlight a given token as an error if it is not contained within the dictionary, and therefore does not validate if a token which is spelt correctly (known to the dictionary) is actually the intended word by the user (i.e. a real-word error). Due to the timeframe available, the author saw fit that time would be better spent creating a high quality and correctly sized dictionary, which aids in combating any potential real-word errors, instead of the implementation of a robust validation component. This can be seen as a weakness of the paper, however the author has taken measures to reduce the potential effect real-word errors could have on results.

This has led to error detection being a very simple dictionary lookup for each token. If a given token is not within the dictionary, then it is marked as an error and corrected.

4.3 Error Correction

This sub-process is the most labour intensive out of the four and concerns generating a correction for a given token after having been highlighted as an *error*. As illustrated in Figure 1, this subprocess can be split again into three smaller processes:

1. Generating candidate words suitable of correcting the given error token.
2. Scoring each candidate against the similarity to the error token and contextual information surrounding the error token.
3. Selecting the highest scoring candidate token to act as the correction for the given error token.

Exploring the task described in point one, the system generates an array of words that it believes to be similar to the error token and that are all known to the dictionary (thus guaranteeing correct spelling). By computing the Soundex[23][57] key of the given error token, the system can select words from the dictionary that possess

the same key and therefore generate words that it believes to be **phonetically** similar to the error token. This process is limited in range however as the phonetically similar word found must have a Levenshtein distance[65] of three or less. Parallel to this process, the system mutates the error token by a single-character **morphological** edit operation (deletion, insertion or substitution) to generate an assortment of variations. Each of these variants is then validated against the dictionary to remove any which the dictionary does not consider to be a word. By combining all unique phonetically and morphologically generated candidates, the system now has a list of words it believes to contain the accurate correction for the error token.

It should be noted that the decision to only generate morphological variants based on an edit distance of one was due to the findings of Section 3. It clearly states that the majority (53%) of errors found were of one edit distance. The remaining 47% of errors are then to be found through phonetic similarity, thus highlighting the reason for a hybrid approach when generating candidate correction words.

Moving on to the second point, the system now goes about computing a score for each candidate of between 0 and 1, where 1 represents the best possible candidate correction word for the error token and 0 represents the worst. This score is calculated using a weighted algorithm (outlined in Section 4.3.1) composed of two main components; the similarity a given candidate has to the error token and the probability of a given candidate being found within the surrounding context of the error token. Again the hybrid approach implemented in this part of the system was a deliberate attempt by the author to widen the analysis made by the system before it attempts to select a candidate token to correct an error with.

As shown in the equation found in Section 4.3.1, the system only utilises bigrams (collection method described in Section 4.5) when calculating the contextual probability that a candidate fits the context of a given error token. By validating both the bigram to the left and right of the error token, the system goes some way to simulate what a trigram may highlight. Due to this the author saw no need to also validate the surrounding trigrams of the error token, as it was seen to be an added complexity. The timeframe available to the author was also taken into consideration when making this decision, as the time and computational effort necessary to collect a high quality set of trigrams in addition to bigrams would exceed what was available.

Finally examining the third point, the system simply ranks each candidate's score in descending order and selects the highest score to act as the correction for the given error token. In the case of no candidates being found the error is dismissed and remains within the original string.

Prior to the execution of this extensive sub-process, the system simply looks up the given error token in a pre-defined social media word correction list. This was created from the most common variants discovered by the linguistic analysis detailed in Section 3 and was amended throughout development when needed. If the given token is present in the word correction list then the respective correction is selected and returned to be substituted into the original string, therefore bypassing the need for candidate generation/selection and in-turn reducing computational load. This is only the case for the most common variants which are guaranteed to only have one meaning (e.g. `'u'` → `'you'`).

4.3.1 Candidate Scoring Algorithm

This section examines the algorithm (shown below) which assigns a numeric value of between 0 and 1 to a given candidate i, as well as a description of how the algorithm was weighted for the highest possible accuracy.

$$\mathbf{C_i} = \frac{w_1\left(\frac{w_3\left(\frac{F_i}{\sum F_c}\right) + w_4(L_i) + w_5(S_i)}{w_3 + w_4 + w_5}\right) + w_2\left(\frac{w_6\left(\frac{F_l}{\sum F_{lb}}\right) + w_7\left(\frac{F_r}{\sum F_{rb}}\right)}{w_6 + w_7}\right)}{w_1 + w_2}$$

Where:

i - candidate currently being scored.

c - all candidates for given error token.

l - candidate left bigram [i, token after error token].

lb - all possible left bigrams from candidates (c).

r - candidate right bigram [token before error token, i].

rb - all possible right bigrams from candidates (c).

F - the frequency for a given token as observed by the dictionary.

L - similarity of string x to string y according to a weighted Levenshtein assessment, between 0-1.

S - similarity of string x to string y according to the longest common substring, between 0-1.

$\mathbf{w_1}$ - [75] overall weighting for similarity part of algorithm.

$\mathbf{w_2}$ - [25] overall weighting for contextual part of algorithm.

$\mathbf{w_3}$ - [33.3] candidate frequency weighting.

$\mathbf{w_4}$ - [33.3] candidate Levenshtein weighting.

$\mathbf{w_5}$ - [33.3] candidate substring weighting.

$\mathbf{w_6}$ - [50] left candidate bigram weighting.

$\mathbf{w_7}$ - [50] right candidate bigram weighting.

The system was initially designed with simplicity in mind and thus did not have the ability to weight each component of the algorithm. The addition of this feature was implemented after the overall system had been completed and thus allowed for each weight to be set through automated testing. Using increments of 5, every possible combination of weights was computed and the output of each was evaluated on a 500 word data set extracted from the manual normalisation of 2,000 YTC carried out for Section 3. This highlighted the best combination of weights, which was then finalised in the system and used for the performance evaluation detailed in Section 5. The execution of this automated evaluation did vastly improve the results, however the limited size of the data set (500 words) could be seen as a limitation, and thus allows for future improvements to be made.

4.4 Correction Substitution

This sub-process deals with replacing a given error with its computed correction. This is simply done by searching the original string for the index of the error word and replacing it with the correction word, starting with the error nearest to the start of the original string (detected first). This is necessary as there may be two or more of the same misspelt error words; however they could have different corrections based on their context, thus it is important to iterate from the start of the original string to the end so that the order is preserved.

4.5 Evolving Dictionary

A point that was repeatedly highlighted throughout the previous literature discussed in Section 2.1.1 was the importance of a high quality appropriately sized dictionary. Fundamentally this is due to the role a dictionary plays in detecting errors within a given string, and consequently the overall accuracy of a normaliser as if errors are incorrectly detected (missed or false-positive) then the task of correcting them becomes purposeless.

Taking this information into consideration, the author saw fit to put into effect two novel features when designing the dictionary for the proposed solution. Foremost the dictionary must be of the correct size, hence the dictionary was populated to be as large as possible and then scoped to be the optimum size. This sizing was quantified through a set of diagnostic tests which evaluated the performance of the system when using different sized dictionaries. Secondly, the need to stop the dictionary from becoming stagnant was met by dynamically

trawling the web for new text from fully formatted sources (e.g. BBC News articles, Wikipedia). This allowed the dictionary to be **dynamic**, making sure to add new words to the dictionary when they are found and to maintain the order of the most common words currently being used on the web. The latter is significant due to the continual reworking of the language considered to be '*common*' on the web.

The process of generating and then maintaining the dictionary with new social content from the web was the same for both creating the word list and a list of bigrams, and each features the same method for preserving the natural state of a given token/phrase (e.g. capitalised).

5 Evaluation

This section of the paper attempts to evaluate the system against each of the two hypotheses outlined in Section 1.3. It will consist of two separate evaluations, each attempting to assess how well independent parts of the system perform in differing environments. The aim and details of each evaluation are described below:

1. Examining the overall accuracy at which the system can correct errors within a corpus of YTC, partnered with a survey of how the quality of text alters after normalisation has taken place.
2. Studying the effect the size and quality of a dictionary has on detecting errors within a corpus of YTC, quantified using precision and recall.

A 2,000 word corpus sourced from 220 randomly selected YTC will be employed in each of the evaluations. It is important to stress that it was completely independent to the implementation of the system and therefore at no point was used to train parts of the system. The corpus was manually normalised by a researcher, thus allowing each evaluation to compare the system's output to what is considered to be the perfect output.

5.1 Accuracy and Text Quality

This evaluation consists of comparing how many of the corrections made by the researcher were also computed automatically by the system. The overall change in text quality will also be assessed through the use of the BLEU metric. The effect each component has on the overall accuracy of the system will also be evaluated by executing the same test over specific parts of the system, thus demonstrating how many successful corrections each component contributes (as seen in Table 8).

Table 8: Comparison of Errors Corrected by each System Component

Component	**Errors Corrected**
Social Media Word List	131 (71.58%)
Phonetic/Morphological Similarity	43 (23.50%)
Repeating Patterns	8 (4.37%)
Bigram Contextual Likelihood	1 (0.55%)

The overall accuracy for the system was 80.26% as it successfully corrected 183 out of the 228 errors found to be in the test corpus by the researcher. It is immediately apparent that the pre-defined social media word list populated from the results of the linguistic analysis played a ***vital*** role in the overall accuracy of the system. This was simply down to allowing the system to swiftly correct the most common errors which are certain to only have one possible meaning (e.g. `luv` → `love`).

The disappointing contribution that the contextual likelihood had to the overall accuracy of the system may be down to the casual nature that YTC feature. As the dictionary was sourced from well formatted web-based sources (e.g. BBC News articles), the system thus relied upon the same contextual behaviour being reflected in

YTC. This may not be the case as YTC often are written in the first person directly offering an opinion, which is very rarely the case in an unbiased news article.

The test corpus displayed a 19.68% overall increase in text quality subsequent to normalisation, thus adding further confirmation of the normaliser's high standard of accuracy. The baseline BLEU metric was calculated to be 0.7983, which was then improved to a value of 0.9554 due to normalisation.

5.2 Error Detection

This evaluation will consist of observing how the precision and recall for error detection within the 2,000 word YTC corpus varies when the system is implemented with dictionaries of varying sizes and quality. Through this exercise the precision and recall for the system itself will also be illustrated. Thus the aim of this is to evaluate how well the system detects errors in YTC when compared to a human and secondly to carry out a novel investigation into the effects a dictionary has on the performance of a normaliser. At this point it is important to stress the fundamental role a dictionary has within a normaliser, as if errors are incorrectly detected then the task of correcting them actually introduces errors into the corpus. Throughout this evaluation the following two-by-two contingency table was used:

Table 9: Precision and Recall Contingency Table

		Manually Classified by Researcher	
		Error	**Not Error**
Automatically Classified by System	**Error**	TP	FP
	Not Error	FN	TN

Where:

True Positive (TP)
The word was classified as a variant by both the researcher and the system.

False Positive (FP)
The system classified the word as a variant however the researcher classed it as a non-variant.

True Negative (TN)
The word was classified as a non-variant by both the researcher and the system.

False Negative (FN)
The system classified the word as a non-variant however the researcher classed it as a variant.

The evaluation will focus on two unrelated dictionaries; the first being the dynamic dictionary implemented within the system (as described in Section 4.5) which sources words from new social content on the web, and the second being a standard static dictionary sourced from the British National Corpus[5]. Both dictionaries had a very similar number of unique words, with the dynamic dictionary (42,838) possessing slightly more than the static dictionary (39,362).

Both dictionaries were ranked in descending order of frequency and split into fifty chunks of words, starting at 1,000 and iterating up to 50,000 at 1,000 word intervals. Each chunk was then used to detect errors within the 2,000 word YTC corpus, subsequently allowing the precision, recall and F_1 measure to be calculated for each chunk. Figure 2 and 3 illustrate how each of the three metric's trend changes as the proportion of words used from the dictionary increases.

[5]Word Frequencies based on the British National Corpus (UCREL): `http://ucrel.lancs.ac.uk/bncfreq/`

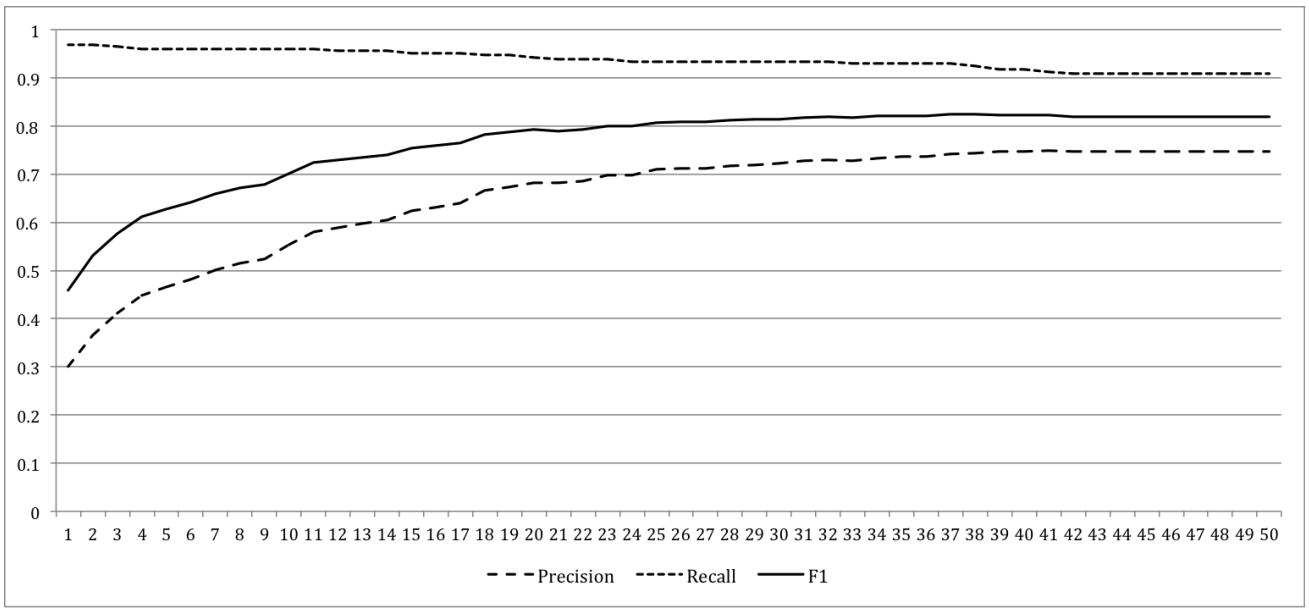

Figure 2: Precision, recall and F_1 measure for the dynamic dictionary.

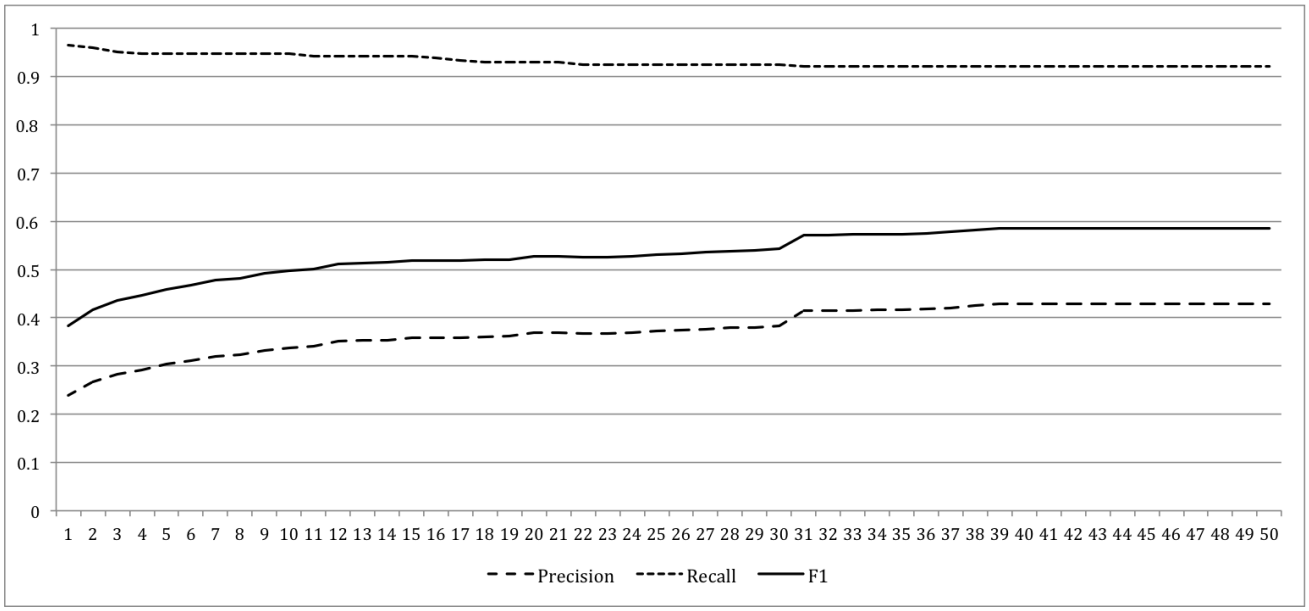

Figure 3: Precision, recall and F_1 measure for the static dictionary.

As shown by Table 10, the overall precision and F_1 measure results for the detection of errors within YTC are significantly higher when utilising the dynamic dictionary compared with the static dictionary, whilst the recall metric remains almost identical across both dictionaries. Furthermore the proportion of the dynamic dictionary that was necessary to be used to obtain the optimum results was just under 13% smaller than what was necessary when using the static dictionary, where almost the entire dictionary was required. This therefore illustrates, with confidence, that the time elapsed in continually sourcing new words from creditable social web-based sources leads to a more reliable dictionary and thus a more robust normaliser.

Table 10: Comparison of Error Detection Accuracy by Dictionary

Dictionary	**Precision**	**Recall**	**F_1**	**Optimum Size**
Dynamic	74.13%	92.98%	82.49%	37,000 words (86.37%)
Static	42.94%	92.11%	58.58%	39,000 words (99.08%)

Examining the false positive and false negative detections made by the dynamic dictionary it is clear that future efforts need to be made to further improve its ability to recognise named-entities specific to a social domain, as throughout the 2,000 word YTC corpus the dictionary incorrectly detected the names of multiple YouTube video creators.

6 Conclusion

This paper set out to prove that a hybrid approach to normalisation can yield a high standard of accuracy when correcting a corpus of YouTube comments. Furthermore, careful attention to the construction and maintenance of the dictionary used by a normaliser increases the overall accuracy of a given system. It accomplished this through implementing the first normaliser able to correct YouTube comments accurately, partnered with a novel dynamic design for a dictionary.

In Section 2 of the paper, the background to text normalisation was discussed at length, paying close attention to traditional methods for spell checking. Section 3 provided a thorough linguistic analysis of YouTube comments, which stressed the importance of this paper by discovering that YTC have the highest level of variance of any social text. Section 4 described how the system was implemented, including the NLP concepts featured and the algorithm created for the purpose of error correction. Section 5 presented two separate quantitative based evaluations addressing the overall accuracy of the system and the effect of using a dynamic dictionary had on error detection rates.

Throughout both of the evaluations carried out the system achieved high accuracy levels ($> 80\%$). Taking this into account, the author believes there is enough evidence to accept both hypotheses and to allow the use of the system within a much larger application allowing YouTube video creators to automatically extract value from their audience's comments through the use of existing NLP tools. This possible future application relies entirely on the effective normalisation of YouTube comments, which the system has shown to be fully capable of completing at a high standard of accuracy.

This paper could be extended with further work in both the linguistic analysis and the evaluation of the application. This paper has highlighted the distinct nature of YouTube comments as a linguistic genre with more spelling variation, and thus a full length linguistic analysis, similar to that carried out on SMS messages by Baron and Rayson[62], is warranted. Secondly, the application would greatly benefit from evaluating the effect it has on the accuracy of existing NLP tools (e.g. Stanford NER) when acting as a pre-processor for variance in social text.

References

[1] *YouTube - Pyropuncher. [Accessed March 2014].* `https://www.youtube.com/user/Pyropuncher`.

[2] *YouTube Statistics. [Accessed Jan 2014].* `http://youtube.com/yt/press/en-GB/statistics.html`.

[3] Ahmad Ammari, Vania Dimitrova, and Dimoklis Despotakis. Semantically enriched machine learning approach to filter YouTube comments for socially augmented user models. *UMAP*, pages 71–85, 2011.

[4] AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. A phrase-based statistical model for SMS text normalization. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 33–40. Association for Computational Linguistics, 2006.

[5] Alistair Baron. Dealing with spelling variation in early modern english texts. *Unpublished thesis, Lancaster University.* 2011.

[6] Alistair Baron and Paul Rayson. Automatic standardisation of texts containing spelling variation: How much training data do you need? 2009.

[7] Alistair Baron, Paul Rayson, and Dawn Archer. Automatic standardization of spelling for historical text mining. 2009.

[8] Amparo Elizabeth Cano Basave, Andrea Varga, Matthew Rowe, Milan Stankovic, and Aba-Sah Dadzie. Making sense of microposts (# msm2013) concept extraction challenge. In *MSM*, volume 13, pages 1–15, 2013.

[9] Richard Beaufort, Sophie Roekhaut, Louise-Amélie Cougnon, and Cédrick Fairon. A hybrid rule/model-based finite-state framework for normalizing SMS messages. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 770–779. Association for Computational Linguistics, 2010.

[10] Jon Bentley. Programming pearls: A spelling checker. *Commun. ACM*, 28(5):456–462, May 1985.

[11] Alexander K Bocast. Method and apparatus for reconstructing a token from a token fragment, April 16 1991. US Patent 5,008,818.

[12] Kalina Bontcheva, Leon Derczynski, Adam Funk, Mark A. Greenwood, Diana Maynard, and Niraj Aswani. TwitIE: An open-source information extraction pipeline for microblog text. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*. Association for Computational Linguistics, 2013.

[13] Kalina Bontcheva and Dominic Rout. Making sense of social media streams through semantics: a survey. *Semantic Web*, 2012.

[14] Chris Callison-Burch and Miles Osborne. Re-evaluating the role of BLEU in machine translation research. In *In EACL*. Citeseer, 2006.

[15] William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. *Ann Arbor MI*, 48113(2):161–175, 1994.

[16] Asli Celikyilmaz, Dilek Hakkani-Tur, and Junlan Feng. Probabilistic model-based sentiment analysis of Twitter messages. In *Spoken Language Technology Workshop (SLT), 2010 IEEE*, pages 79–84. IEEE, 2010.

[17] Xu Cheng, Cameron Dale, and Jiangchuan Liu. Statistics and social network of YouTube videos. In *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pages 229–238. IEEE, 2008.

[18] Kenneth W Church and William A Gale. Probability scoring for spelling correction. *Statistics and Computing*, 1(2):93–103, 1991.

[19] Eleanor Clark and Kenji Araki. Text normalization in social media: Progress, problems and applications for a pre-processing system of casual english. *Procedia - Social and Behavioral Sciences*, 27(0):2 – 11, 2011. Computational Linguistics and Related Fields.

[20] Paul Cook and Suzanne Stevenson. An unsupervised model for text message normalization. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 71–78. Association for Computational Linguistics, 2009.

[21] Fred J Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.

[22] Fred J. Damerau. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176, March 1964.

[23] Leon Davidson. Retrieval of misspelled names in an airlines passenger record system. *Communications of the ACM*, 5(3):169–171, 1962.

[24] Etienne Denoual and Yves Lepage. BLEU in characters: towards automatic mt evaluation in languages without word delimiters. In *Companion Volume to the Proceedings of the Second International Joint Conference on Natural Language Processing*, pages 81–86, 2005.

[25] Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, 2013.

[26] Ivor Durham, David A. Lamb, and James B. Saxe. Spelling correction in user interfaces. *Commun. ACM*, 26(10):764–773, October 1983.

[27] Edward A Fox, Qi Fan Chen, and Lenwood S Heath. A faster algorithm for constructing minimal perfect hash functions. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 266–273. ACM, 1992.

[28] RE Gorin. SPELL: A spelling checking and correction program. *Online documentation for the DEC-10 computer*, 1971.

[29] Stephan Gouws, Dirk Hovy, and Donald Metzler. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pages 82–90. Association for Computational Linguistics, 2011.

[30] Bo Han and Timothy Baldwin. Lexical normalisation of short text messages: Makn sens a# Twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 368–378. Association for Computational Linguistics, 2011.

[31] Bo Han, Paul Cook, and Timothy Baldwin. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 421–432. Association for Computational Linguistics, 2012.

[32] Leon D Harmon. Automatic recognition of print and script. *Proceedings of the IEEE*, 60(10):1165–1176, 1972.

[33] George E. Heidorn, Karen Jensen, Lance A. Miller, Roy J. Byrd, and Martin S Chodorow. The EPISTLE text-critiquing system. *IBM Systems journal*, 21(3):305–326, 1982.

[34] Max Kaufmann and Jugal Kalita. Syntactic normalization of Twitter messages. In *International conference on natural language processing, Kharagpur, India*, 2010.

[35] Gerard Kempen and Theo Vosse. A language-sensitive text editor for dutch. In *Computers and Writing*, pages 68–77. Springer, 1992.

[36] Catherine Kobus, François Yvon, and Géraldine Damnati. Normalizing SMS: are two metaphors better than one? In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 441–448. Association for Computational Linguistics, 2008.

[37] Karen Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 24(4):377–439, 1992.

[38] Amanda Lenhart, Kristen Purcell, Aaron Smith, and Kathryn Zickuhr. *Social media & mobile internet use among teens and young adults*. Pew internet & american life project Washington, DC, 2010.

[39] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966.

[40] Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. Insertion, deletion, or substitution? normalizing text messages without pre-categorization nor supervision. *ACL (Short Papers)*, 11:71–76, 2011.

[41] Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. Recognizing named entities in tweets. In *ACL*, pages 359–367, 2011.

[42] Eric Mays, Fred J Damerau, and Robert L Mercer. Context based spelling correction. *Information Processing & Management*, 27(5):517–522, 1991.

[43] M McIlroy. Development of a spelling list. *Communications, IEEE Transactions on*, 30(1):91–99, 1982.

[44] Roger Mitton. Spelling checkers, spelling correctors and the misspellings of poor spellers. *Information processing & management*, 23(5):495–505, 1987.

[45] Roger Mitton. Ordering the suggestions of a spellchecker without using context. *Natural Language Engineering*, 15(2):173–192, 2009.

[46] Robert Morris and Lorinda L Cherry. Computer detection of typographical errors. *Professional Communication, IEEE Transactions on*, (1):54–56, 1975.

[47] Robert Nix. Experience with a space efficient way to store a dictionary. *Communications of the ACM*, 24(5):297–298, 1981.

[48] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[49] Peter K Pearson. Fast hashing of variable-length text strings. *Communications of the ACM*, 33(6):677–680, 1990.

[50] Jennifer Pedler and Roger Mitton. A large list of confusion sets for spellchecking assessed against a corpus of real-word errors. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10)*, 2010.

[51] James L Peterson. Computer programs for detecting and correcting spelling errors. *Communications of the ACM*, 23(12):676–687, 1980.

[52] James L Peterson. A note on undetected typing errors. *Communications of the ACM*, 29(7):633–637, 1986.

[53] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. The Edinburgh Twitter corpus. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, pages 25–26, 2010.

[54] Joseph J Pollock and Antonio Zamora. Automatic spelling correction in scientific and scholarly text. *Communications of the ACM*, 27(4):358–368, 1984.

[55] Edward M Riseman and Allen R Hanson. A contextual postprocessing system for error correction using binary n-grams. *Computers, IEEE Transactions on*, 100(5):480–493, 1974.

[56] Alan Ritter, Sam Clark, Oren Etzioni, et al. Named entity recognition in Tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics, 2011.

[57] RC Russell and KM Odell. Soundex phonetic comparison system [cf. us patents 1261167 (1918), 1435663 (1922)], 1922.

[58] Peter Schultes, Verena Dorner, and Franz Lehner. Leave a comment! an in-depth analysis of user comments on YouTube. In *Wirtschaftsinformatik*, page 42, 2013.

[59] Stefan Siersdorfer, Sergiu Chelaru, Wolfgang Nejdl, and Jose San Pedro. How useful are your comments?: Analyzing and predicting YouTube comments and comment ratings. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 891–900, New York, NY, USA, 2010. ACM.

[60] L Venkata Subramaniam, Shourya Roy, Tanveer A Faruquie, and Sumit Negi. A survey of types of text noise and techniques to handle noisy text. In *Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, pages 115–122. ACM, 2009.

[61] Caroline Tagg. *A corpus linguistics study of SMS text messaging*. PhD thesis, The University of Birmingham, 2009.

[62] Caroline Tagg, Alistair Baron, and Paul Rayson. i didnt spel that wrong did i. oops: Analysis and normalisation of SMS spelling variation. *Lingvisticæ Investigationes*, 35(2):367–388, 2012.

[63] P Tenczar and W Golden. Cerl report x-35. *Computer-Based Educatmn Research Lab., Umv of Ilhnois, Urbana, Ill*, 1972.

[64] Thomas N. Turba. Checking for spelling and typographical errors in computer-based text. In *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation*, pages 51–60, New York, NY, USA, 1981. ACM.

[65] Robert A Wagner and Michael J Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, 1974.

[66] Donald Walker and Robert Amsler. The use of machine-readable dictionaries in sublanguage analysis. *Analyzing Language in Restricted Domains*, pages 69–83, 1986.

[67] Emmanuel J Yannakoudakis and D Fawthrop. An intelligent spelling error corrector. *Information Processing & Management*, 19(2):101–108, 1983.

[68] Ying Zhang, Stephan Vogel, and Alex Waibel. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system? In *LREC*, 2004.