UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Final Paper 2020

Title: Building a South African English corpus

Author: M.Umar Khan (KHNMUH038)

Project Abbreviation: CASE TEXT

Supervisor(s): Associate Professor Maria Keet

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	15
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	0
System Development and Implementation	*0*	*20*	20
Results, Findings and Conclusions	*10*	*20*	10
Aim Formulation and Background Work	*10*	*15*	15
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	

Building a South African English corpus

Muhammed Umar Khan
khnmuh038@myuct.ac.za
University of Cape Town
Rondebosch, South Africa

Abstract

A corpus is a compilation of text that is used for linguistic research. Currently, South African English(SAE) does not have a written electronic corpus and only has smaller spoken corpora that only focus on the sub-varieties of SAE. This project aimed to create a comprehensive corpus for SAE which would fairly represent multiple sub-varieties of the language. The corpus was created by gathering, cleaning, and storing data from multiple different categories where SAE is used. After completing the development of the corpus, we achieved to develop an electronic South African English with a comprehensive metadata scheme for the contents of the corpus. Weaknesses of the corpus were discussed (such as an unfair representation of sub-varieties of SAE) and recommendations for future works were provided on how the corpus could be expanded.

Keywords

Corpus, South African English(SAE)

1 Introduction

A corpus is a collection of texts which are used for linguistic research. South African English (SAE) contains many sub-varieties which include: Indian South African English, Black South African English, Afrikaans South African English, etc. South African English has no large-scale electronic corpus available, so linguistic research is limited due to there being no electronic corpus being available for the language [1].

A South African English corpus is needed for projects such as the International corpus of English or ICE requires that there be a South African English corpus as it is one of the components of the ICE [5].The need for a South African English corpus is further enhanced by the fact that an authority of SAE, the Dictionary of South African English (DSAE), requires that an SAE corpus be available because their current approach of manually collecting data to compile SAE dictionaries seem to limit the amount of data that can be gathered and also limits the depth to which the analysis of data can take place[1]. This project doesn't aim to fulfill those requirements but the requirements presented by these organizations facilitate the need for an SAE corpus for linguistic research.

Previously there has been work done in developing corpora for South African English but these corpora focused on sub-varieties of South African English such as Black South African English and Indian South African English [1]. The data samples for these corpora were relatively small in size as data was collected the use of audio recordings [3, 5, 4] rendering these corpora not useful for projects with a large scale and involving written text, these corpora also did not take into account various genres where South African English is used such as news websites, political speeches, blogs, social media and so on [5, 4].

Taking all these factors into consideration this project was undertaken to fulfill the following aims:

- Build a corpus for South African English that could easily be expanded upon in the future and be capable to use for linguistic research by the end of this project.
- Development of an appropriate metadata scheme for the corpus, which will help in adding linguistic context and provenance to the corpus and make it simpler to traverse through the corpus and extract information from it.
- Attempt to fairly represent most sub-varieties of South African English in the corpus.

The development of included 4 major steps with Testing being done between steps. These 4 steps were as follows:

(1) Gathering data for the corpus
(2) Cleaning and Filtering of raw data
(3) Implementing Metadata of the corpus
(4) Storage of data acquired

This paper consists of 8 sections. In Section 2 we discuss the related work done concerning developing a South African corpus as well as the methods that were used during the development of other language-specific corpora. Section 3 covers the Design. Section 4 covers the implementation of the design for the development of the corpus and section 5 details the testing strategy. Section 6 details the results we achieved as well as the discussion of the results. Section 7 is further discussion of key results. Section 8 covers the conclusion and the possibility of any future work.

2 Related works

In this section we will be focusing on the related work that has been done for developing corpora for South African English as well as developing various language-specific corpora worldwide, the methods used for developing these corpora allowed us to choose the tools appropriate for the development of the South African English corpus.

2.1 Previous Work on the Development of SAE Corpora

Currently, there exists no complete corpus for South African English [1, 3], work has been done on the sub-varieties of South African English this includes work done by Peinaar and de Klerk [3] concerning Indian South African English, de Klerk [4] for Xhosa English in which De Klerk came up with a spoken corpus of Xhosa English consisting of about 500000 words [4] and Jeffery [5] with an attempt to create a South African English corpus for the International Corpus of English, but this corpus has not yet been completed. A trait common between these corpora was that they only focused on sub-varieties of South African English and their means of collecting data for the corpus was limited to audio recordings as seen with [3], [4] and [5] this resulted in a very small sample size [1].

The decision to use audio recording was explained by [3], since to transform from spoken data to written text required listening to the recordings multiple times and manual transcription of the words. It was acknowledged by [3] that automatic transcription methods do exist but the issue with those is that they tend to perform well with clear speech such as dialogues or broadcast monologues, whilst with unpredictable and spontaneous speech auto transcribers are still unreliable [3].

2.2 Work on other language-specific corpora

Generally, the following procedure was adopted for the development of corpora by various scholars.

(1) Collecting raw corpus data
(2) Storage of acquired corpus data
(3) Filtering and cleaning acquired data
(4) Metadata scheme for the corpus

2.2.1 **Collecting raw corpus data** [1] specifically worked on developing a corpus for South African English and the methods used by [1] were the use of web crawlers, RSS feeds and software to collect dynamic data such as CrawlJax.

[6] and [2] which constructed large corpora used web crawls as well but also highlighted the possible use of methods such as search engine queries and search engine hit counts.

2.2.2 **Storage of data acquired** Most scholars did not state how the corpus was stored [1].[1] used NoSQL databases such as MongoDB to store the data that was retrieved for the corpus and scholars such as [8] chose to store data with the use of text files as they believed this approach was quicker than making the use of relational databases.

2.2.3 **Cleaning and filtering data** After the corpus data was collected it required filtering and cleaning, procedures such as boilerplate removal had to be carried out by [1, 2, 6] and either developed or used specialized tools for this process such as Boilerpipe, NCleaner,Recovery etc.

The BTE tool was adopted by [12] in which it was stated that sections rich with content in a page would often have a low HTML tag density, whereas boilerplate contains a substantial amount of HTML due to its special formatting, newlines etc .This approach was independent of the crawling procedure used because it is based on the general properties of web documents [10].

[9] decided to go with manually writing scripts for each website to deal with HTML markup and boilerplate removal, the reason for this was because they believed this method would result in desired content and would save them from the problem of fundamental duplicates such as samples from articles and blogs etc [9].

Automatic cleaning tools were also developed by some, such as "Victor the Cleaner" by [11] which was aimed at cleaning HTML pages by removing all text except headers and main page content. Continuous text sections (sections not including any HTML tags) are considered a single block that should be marked by a label as a whole [11].

Whilst carrying out the cleaning process scholars also performed the removal of near duplicates or exact duplicates of data. Exact duplicates were easier to handle by using hashes and fingerprints to track HTML documents [1]. Near duplicates, removal was a harder task but the general approach seen with many such as [1], [2] and [6] was the use of n-grams to track the similarity between documents and also implemented an algorithm known as the "shingling" algorithm to deal with near duplicates [1, 6, 2].

2.2.4 **Metadata scheme for the corpus** [7] suggested the use of XML files and metadata schemes such as Text Encoding Initiative (TEI),ISLE Metadata Initiative (IMDI), and the Open Language Archive Community (OLAC). These schemas can categorize any resource and TEI specifically has a detailed online guideline.

Most of the schemas were very similar when categorizing data but with the TEI approach, custom schemas could be made by extending the TEI corpus schema. If this was done usually a document was provided detailing the definitions of all the elements used in the schema [7].

Most authors did not mention the way they stored the metadata but there are versions of the Brown Corpus as well as the BNC that are available in XML and use TEI as their metadata schema [7].

3 Design

In this section, we discuss how the requirements for the project were gathered and the design aspects of the corpus.

3.1 Requirements Gathering

To gather the requirements of the project Weekly meetings on Skype were scheduled with the project supervisor. These meetings were also kept to prevent any scope creep during the course of the project. Any further doubts with regards to project requirements were clarified by the project supervisor over email. The requirements were as follows:

- Attempt to make an SAE corpus over a million tokens
- Include various sub-categories of data for the corpus with data belonging to different provinces of South Africa to attempt and fairly represent sub-varieties of SAE.
- Implement a metadata scheme that would make it easy to locate the source of the text that was gathered and make it easy to traverse through the corpus.

With these requirements in mind, we came up with the design of our corpus.

3.2 System Design

The corpus was designed to contain over a million tokens of data, comprising of various genres where South African English was used, the reason for this was because different words and style of writing is probable in different forms of media, for example, a newspaper article is expected to be more professional than a simple social media tweet. Because of this, we decided to include the following genres for the corpus:

(1) News websites
(2) Political speeches
(3) Media statements and Advisories
(4) South African Books(Fiction)
(5) South African Twitter data
(6) South African Blogs

For some categories of data, we decided to include sub-categories, the reason for this choice was that there could be a difference in the writing style between the sub-categories.

A list of the subcategories are as follows:
For news websites:

(1) National News
(2) Local News
(3) Lifestyle and Entertainment
(4) Sport
(5) Health
(6) Education

For Blogs :

(1) Lifestyle
(2) Travel

For Books :

(1) South African fiction

We also decided to gather text from various regions of South Africa such as Gauteng, Western Cape, KwaZulu-Natal, etc. This choice was made to attempt to fairly represent different sub-varieties of South African English such as Indian South African English(containing words such as "Whakind" as a form of a greeting), Black South African English (which contains words like "indaba" meaning meeting of a community, or words such as "ubuntu" used for kindness), Afrikaans South African English(comprising of words such as "boykie" for a young man) and so on. These sub-varieties were included as South African English comprises of many "loan" words from such languages [1].

Sub-category	Region	Example where found
National News	National	News24
Local News	KwaZulu-Natal	The Mercury
Sport (News)	Eastern Cape	HeraldLive
Health (News)	National	Mail and guardian
Education (News)	National	Mail and guardian
Lifestyle and Entertainment (News)	Western Cape	Daily Voice
Finance (News)	National	The South African
Travel (Blogs)	Gauteng	Rattle and Mum
Lifestyle (Blogs)	Western Cape	Boring Cape Town Chick
South African fiction (Books)	National	Project Gutenberg

Figure 1: Example showing the sub categories of the data that will be gathered along with examples where such sub-categories were represented

Metadata is essential when it comes to organizing ways a language specific corpus can be meaningfully processed [7]. It stores the interpretive framework in which the constituents of a corpus are selected and are to be understood [7]. Cause of these reasons metadata is vital to a language corpus. This is why we came up with a comprehensive metadata scheme for our corpus.

We decided to store the corpus in two different ways one in XML format and the other in flat-text format. The reason this was done is for the XML version it used the TEI corpus

metadata schema which is a standard metadata schema and can be used with tools that cater to it. We also chose to do a flat-text version of the corpus with our custom metadata schema, this version was chosen as not all corpus tools such as WordSmithTools [14] and AntConc[15] are comfortable with XML format of data and the flat-text version can be easier to use when dealing with just the contents of the corpus as they are not encoded with any tags like with XML.

The TEI corpus metadata schema was intended to be used for the XML version of the corpus and our custom metadata schema for the flat-text version of the corpus. Both the metadata schemas contained the same contents albeit in a different format the metadata for both versions of the corpus included the following:

- Title
- author
- publisher
- Date published
- Country
- Province/ region(in XML)
- Source URL
- Category
- Sub-Category
- License

3.3 Design Approach for the development life cycle

To carry out the development of the corpus 2 major stages need to be carried out before the metadata can be implemented and the storing of the corpus, these stages are as follows:

(1) Gathering raw corpus data
(2) Cleaning and filtering of raw data

To begin with the development of the corpus we first had to decide on how the data would be gathered. Various methods were considered in deciding the methods that were going to be used to gather data for the corpus such as web crawlers, search engine queries and search engine hit counts.

Web crawlers were chosen for data collection. Web crawlers were chosen cause they are efficient at gathering data and many powerful web crawlers are available for free, web crawlers also don't have limitations that are present with methods such as search engine queries, where the number of queries per day is restricted by search engines [2]. This required us to use popular Web crawling packages such as Scrapy and develop Crawlers to suit extraction of data from the sources of our choice such as various news websites or political speeches, this meant that using the Scrapy classes we need to create our custom crawlers to extract data.
The Scrapy based crawlers were designed to be used with all the corpus categories besides the gathering of twitter data which required a separate program be written using the Twitter API as twitter tweets do not have their separate URLs like blog articles or news articles.

When it came to gathering data for the Books category of the corpus it was very difficult to find any recent books which were available for free, this led to us deciding to gather our books data from www.gutenberg.org, here we were able to find books on South African fiction written by South African authors and these books were freely available with almost no restrictions whatsoever, the only restriction was to include the Project Gutenberg license with the book.

When crawling data from the web, the data is still not ideal for corpus development and includes boilerplate as our crawler was made to extract HTML tags as well as the HTML body content. This decision was made because if only the text of specific tags was extracted there could be a loss in valuable data if those tags contained HTML tags that contained other text. An example of this is given in Figure 1:

```
<p>
  "A Lotus River drug "
  <em id="emphasis-80c4c96400adfc4f98e1b63a7bc27958">
  mert</em> == $0
  ", accused of supplying drugs to Mitchells Plain
  gangs, is behind bars after the Anti-Gang Unit (AGU)
  busted him with tik worth over R15 000."
</p>
```

Figure 2: Example showing <p> tags in a Daily voice news article contain tags for South African loan words(in this case "mert" is the loan word)

The cleaning of data was also required to take into account the possibility of data duplication, this was due to the nature of Twitter data and how there is a possibility of retweets, this had to be handled with the use of custom programs as to not store any retweets. Data duplication was also a possibility when conducting large crawls so this had to be handled by not storing any article that had already been crawled before.

3.4 Corpus Development Methodology

An iterative approach was used for the development of the corpus in which initially a small mini corpus was to be developed comprising of only a few news articles along with some basic metadata such as title, author, and date published. An iterative approach was selected for this project as we believe any problems with our current design or procedures could be detected early on in the development life cycle and changes could be easily made in due time, following a waterfall approach we would have possibly resulted in making changes late during the development life cycle and risk not having completed the project in time.

4 Implementation

In this section, we discuss how the design aspects were implemented to develop a South African English corpus.

Below we detail how the design aspects were implemented for the development life cycle.

4.1 Gathering raw corpus data

For gathering data for the corpus we included data from the following sources.

For news websites:

- Mail and guardian
- News24
- The South African
- The Citizen (Gauteng and National news)
- Sunday Times (TimesLive)
- Daily Voice (Western cape news)
- South coast Sun and The Mercury (KwaZulu-Natal news)
- HeraldLive (Eastern Cape news)

For Blogs we gathered data from the following websites :

- Rattle and Mum
- Boring Cape Town Chick
- Minkys
- Dirty Pink City
- Unfold Durban
- Presitge Digital

For political speeches and media statements and advisories, we gathered data from the Government of South Africa's website.

Books were gathered from www.gutenberg.org and Twitter data was gathered from Twitter using Twitters API

The Scrapy crawler which is a python based crawler was one of the methods used for gathering data for the corpus, it was suitable for our needs as Scrapy allows for the easy creation of filters for the cleaning of data as well as data sanitization [1]. These filters were used to extract important metadata for the text being crawled and will be discussed in further detail later on.

For the crawling of data we wrote two separate Python programs to initiate crawls depending on the source, A BaseSpider was designed to carry out crawls where the URLs were fed into the python program and smaller crawls were to be made, and a complex crawler(known as CrawlSpider in Scrapy) was designed when larger crawls were needed and the spider had to follow a base URL depending on some set rules such as crawling articles that contain "lifestyle" in their URL, so this meant that articles part of the lifestyle category may be crawled. The Crawl spider was capped to a specific number of articles(depending on the source) to avoid the Crawler running for a very long time and able to manage the amount of content being crawled.

```
>scrapy crawl large
```

Figure 3: Command used to initiate a crawl in the python terminal (large is the name of the CrawlSpider)

```
start_urls = ['https://www.dailyvoice.co.za/news/-princess-vlei-is-declared-a-heritage-site-
base_url = 'https://www.dailyvoice.co.za/news/'
rules = [Rule(LinkExtractor(allow=("https://www.dailyvoice.co.za/news/",)),
             callback='parse', follow=True, process_links=lambda l: l[:100])]
```

Figure 4: example showing start URL,base URL and follow URL for CrawlSpider crawler

A python script was written using the Twitter API aimed to gather tweets by the use of hashtags such as #Mzansi, #Jozi which were terms that are popularly used by South Africans making the tweets gathered being South African English. To gather data from various regions of South Africa to represent the sub-varieties of SAE, popular cities of a province and the province name were included in the hashtags such as for the Western Cape the hashtags were #WesternCape, #CapeTown. When extracting twitter data the crawls were limited to a max of 1500 raw unfiltered tweets in the past 6 months. The tweets were limited to the past 6 months to try and prevent any tweets posted by tourists during the holiday season at the start of the year.

We did not have to write any custom scripts when gathering book data as Project Gutenburg offered flat-text versions of the books so data was already in its desired format with only the removal of the table of contents and the index being done manually.

4.2 Cleaning and filtering of raw data

Our approach for the removal of boilerplate was done by using an open-source Python library known as jusText, the code written with this library attempted to remove boilerplate from the crawled web pages and produce the main body text for the webpages. The data produced after running the jusText code was the content to be included in the corpus. As mentioned in the previous subsection, twitter tweets were not crawled as they did not contain any HTML code when the raw data was extracted, due to this being the case we were not able to use jusText to remove any boilerplate such as links and hashtags being present in tweets. To remove tweet boilerplate we wrote down python code to remove the links, # and @ from tweets, making the tweets then boilerplate free.

Another issue that was to be taken into account was with large crawls of data, as the Scrapy spider might revisit a page due to it containing the base URL required by the python spider. To counter this issue, when conducting large crawls we stored the title of each crawled article in a list, this list was used to check if an article with the same heading had been crawled previously, and if so then the article data was disregarded. Twitter data also raised the concerns for data duplication as twitter has the option to retweet tweets. This was simple to handle, as retweets contain "@RT" and because of this our program was able to declare a tweet as a retweet and avoid unnecessary data duplication.

```
if (new_string in tweets):# detect any exact duplicates for tweets
    pass
elif (new_string.find("RT")!=-1):# detect retweets
    print("Retweet")
    retweet+=1
else:
    tweets.append(new_string) # Add to the list containing all the tweets
    w.write(new_string + '\n') # write tweet to file
```

Figure 5: Code for handling twitter retweets

4.3 Metadata for the corpus text

Some of the metadata for the corpus was automated with the use of python code, this was done by using a program called SelectorGadget [13] which is a Google Chrome extension which enables to select parts of a webpage and give the Xpath code for how the selected text is stored, using this program we extracted metadata such as article title, author, date published, category of data and so on.

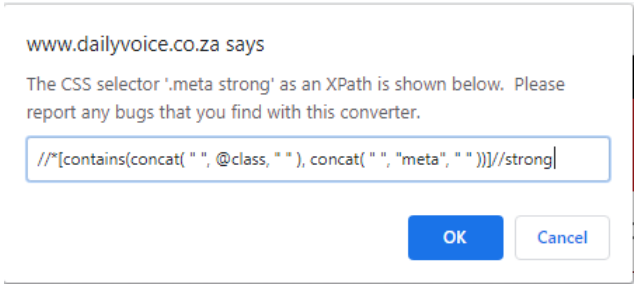

Figure 6: Demo of how the XPath for an author from a daily voice article was extracted using SelectorGadget the author is clicked on the webpage to get the XPath for the author

```
author = response.xpath(
    '//*[contains(concat( " ", @class, " " ), concat( " ", "meta", " " ))]//strong/text()'
).extract()
```

Figure 7: Example of how the Xpath in Figure 6 was used to extract and store the author name in the code

Extraction of metadata varied from source to source as categories such as media statements and Political speeches included data like province which was not present in sources such as news articles or blogs, though this metadata was easily added manually according to the type of data being crawled such as inserting "Lifestyle and Entertainment" under sub-category for crawling pages with the Base URL containing "lifestyle" before writing the content to a flat-text file.

```
w.write("Title: " + title[0].strip() + '\n')
w.write("author: "+ author[0].strip() + '\n')
w.write("publisher: Sunday Times " + '\n')
w.write("Date published: "+ date[0].strip()  + '\n')
w.write("Country: South Africa" + '\n')
w.write("Province: National" + '\n')
w.write("Source URL: " + '\n')
w.write("Category: News" + '\n')
w.write("Sub-Category: Finance"+ '\n')
w.write("License: Text is protected by Copyright 2020 Blue Sky
```

Figure 8: Example showing how automated metadata was added and how some known data was manually inserted before storing the corpus text

Figure 8 shows how for news websites, the publisher, country, province, subcategory, and license were manually inserted before writing data to the corpus for storage.

For the flat-text file implementation of metadata, the start of the metadata was denoted by three #'s followed by the metadata than a single hash at the end to denote the end of the metadata and the start of the text content.

```
#
#
#
Title:
author:
publisher:
Date published:
Country:
Province:
Source URL:
Category:
Sub-Category:
License:
#
text goes here..
```

Figure 9: How metadata was represented in the flat-text file

4.4 Corpus data Storage

As previously mentioned in our design two versions of the corpus were to be developed. For this to be possible we wrote the data after it had been cleaned by jusText to a flat-text file along with adding the metadata as well. Once the flat-text version of the corpus was available we then developed a python program to convert our flat-text variation of the corpus to an XML one, this program provided the TEI representation of the texts and then later, was manually made to

include the <teicorpus> tag as the root element to encompass all the text of the corpus and be following the TEI standards for a corpus.

4.5 Corpus Structure

Corpus data was put into different folders as that it would make it easier to access data belonging to a specific category instead of having to search the corpus to find data belonging to the desired category. A concern that might be raised is that a text might belong to more than one category of data such as it could be a political speech as well as a news article. For this reason, we decided to separate the data into its various categories and also have a separate file that contained all the contents of the corpus, having this large file would make it easier to test if there was any overlap between data categories.

Statistics such as the number of texts and the number of tokens per category of the final corpus are discussed in the Results and Discussion section .

5 Testing strategy

We ran tests for each major step in the development cycle of the corpus. The purpose of these tests was to see that if the corpus could be easily searched and if the assumptions being made during the development process were true.

5.1 Tests for gathering and cleaning data

We initially ran tests for gathering data by writing a short python script and created a simple crawler with Scrapy in which the URLs for the webpages that were intended to be crawled were manually fed into the program. This test was done to monitor the format of the scraped pages so that it may be fed into the jusText code for the removal of boilerplate.

After running the Basic crawler with a few articles from news websites it was seen that data was indeed being crawled successfully and it was noted that the relevant data needed for the corpus was stored in the <p> tags of the web pages and this was extracted and compared with text produced by jusText.This test was done to check the reliability of jusText when attempting to remove boilerplate.

For gathering twitter data we suspected that when attempting to crawl data from different provinces of South Africa that more popular provinces will see a higher number of tweets when compared to provinces that are less developed or have less access to the internet, for this reason when we fixed the number of raw unfiltered tweets to a max of 1500 in the past 6 months and attempted to gather twitter data by inserting hashtags that contained the province name and popular cities of the province.

After the data was gathered, cleaned, and stored we ran it through a python program we wrote to determine the word count of the corpus, this was done so that we could determine if we reached our goal of over 1 million tokens for the corpus.

5.2 Tests of the metadata to search the corpus

When the corpus data was gathered from a specific source we searched through the contents of the flat-text file using a python program we wrote to determine if data can be retrieved from the corpus with the use of the metadata this meant for example looking for texts in the news websites section that belonged to the sub-category "Lifestyle and Entertainment" or extracting texts that were intended for a specific province such as the "Western Cape".

6 Results and Discussions

When crawling the data we noticed that the CrawlSpider was not feasible for every source, we noted that when inserting a start URL and follow URL for some websites the web crawler was unable to follow and extract other articles that contained the follow URL as part of their URL for example when browsing a news website known as The Citizen we saw that the web crawler crawled the article which we had included as a start URL for the web crawler and fail to crawl most other articles that contained the follow URL, this meant that if we were trying to extract for example some "Finance" articles and fed the URL for one Finance article as the start URL, the crawler would exit only after crawling a few articles. For this reason, for such sources we had to resort to the BaseSpider in which we manually fed the URLs for the webpages that we chose to be extracted, the BaseSpider was also preferred in some instances due to the Scrapy BaseSpider class being able to track the URL of the pages it was crawling, so this meant that the source URL was automated and included as part of the metadata. Concerning the large crawler(CrawlSpider) the source URL had to be manually inserted after the corpus data was stored.

An issue we encountered when using the large crawler was that for websites that had some sections that required a paywall such as news24 our large crawler only managed to crawl very little of the article and because of this jusText ended up including some boilerplate which was repeated throughout "subscriber-only" articles. So in such a case, we either used the BaseSpider program so that we could manually feed URLs for free articles or after storage search all articles containing the "subscriber-only" article boilerplate and removing them manually.

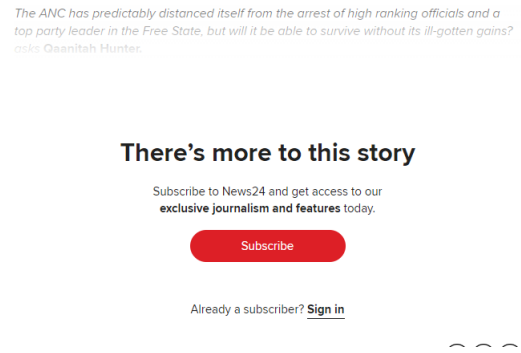

Figure 10: Example of a paid article that the large crawler attempted to crawl

Our results when performing boilerplate removal with jusText were quite satisfactory for most of the sources we tried to crawl we got boilerplate with the aforementioned subscriber-only articles and inconsistencies with main content detection with some other sources. The other sources where we had inconsistencies with jusText were not a big issue as data extracted by jusText from these sources did not include any boilerplate but they did not encompass all the content of the source as well, this was not seen as a major issue as we were gathering fewer data from a source but we were still gathering important text that belonged to that article and not some common boilerplate.

```
Title: Teen beauty queen scoops another pageant crown
author: Venecia Valentine
publisher: Daily Voice
Date published: September 17, 2020
Country: South Africa
Province: Western Cape
Source URL: https://www.dailyvoice.co.za/lifestyle-entertainment/lifestyle/teen-beauty-queen-scoops-another-
Category: News
Sub-Category: Lifestyle and Entertainment
License: Independent Media and affiliated companies. All rights reserved.
#
"Her coach wanted to know why she isn't attending modelling class anymore and I told them we could no longe
Apple, who is in Grade 8 at Mountview High School in Hanover Park, also got to meet one of her heroes, Miss
HIGHLIGHT: Apple and Zozibini Tunzi
"This was one of the highlights for me, it was so amazing to have met Zozibini when Mayor Dan Plato invited
Apple has been nominated for the South Africa Children Awards, taking place December, for her charity work.
#
#
#
```

Figure 11: Example of how jusText only detected a small section of a daily voice "Lifestyle and Entertainment" article

When gathering twitter data by using hashtags from multiple provinces and cities we noted that even though we had set a max limit for tweets (1500 raw tweets including retweets and boilerplate) we noted that some provinces contained much lower amounts of tweets compared to other provinces.

twitterEasternCape.txt	9/23/2020 1:20 PM	Text Document	26 KB
twitterJoburg.txt	9/23/2020 12:23 PM	Text Document	59 KB
twitterKZN.txt	9/23/2020 12:44 PM	Text Document	49 KB
twitterLimpopo.txt	9/23/2020 1:07 PM	Text Document	24 KB
twitterMpumalanga.txt	9/23/2020 2:31 PM	Text Document	14 KB
twitterMzansi.txt	9/26/2020 1:16 PM	Text Document	37 KB
twitterNorthernCape.txt	9/23/2020 2:17 PM	Text Document	3 KB
twitterNorthWest.txt	9/23/2020 2:22 PM	Text Document	4 KB
twitterWesternCape.txt	9/23/2020 12:50 PM	Text Document	73 KB

Figure 12: From the size of the text files we can see how the number of tweets varied between provinces, cities and other keywords such as Mzansi

With regards to twitter data, we also noticed that many slang terms were being used in tweets and some tweets were not in English or were half in English and half in a native language such as Afrikaans. We did not try to change this as attempting to remove the "non-English" words would have potentially resulted in a loss of loan words that are commonly used in SAE.

```
I think it depends where youre at cz we got ours from
Ke kgathetse fuu! King Edward,willows.Bloemfontein ,
Bloemfontein mfanaka
Members of Area Military Health Unit Free State doing
Bloemfontein, yoh
THIS SATURDAY IN BLOEMFONTEIN... Bloemfontein, Free S
```

Figure 13: Shows some tweets were not in english (Seen in the second line of the figure), example was taken from twitter data from the Free state provinces

After gathering, cleaning and storing data our metadata was stored in the following way (Seen in figure 14 and 15) for all categories of data:

```
#
#
#
Title: South Africa goes to Covid-19 level 2 on Monday
author: Lester Kiewit, Sipho Kings
publisher: Mail and Gaurdian
Date published: 15 August 2020
Country: South Africa
Province: National
Source URL: https://mg.co.za/coronavirus-essentials/2020-08-15-south-africa-goes
Category: News
Sub-Category: Health
License: Text is protected by Copyright Mail and Gaurdian. All rights reserved.
#
```

Figure 14: Shows metadata for an article from the Mail and gaurdian in the news websites section

```
    <sourceDesc>

        <bibl>
            <title>South Africa goes to Covid-19 level 2 on Monday</title>
            <author>Lester Kiewit, Sipho Kings</author>
            <publisher>Mail and Gaurdian</publisher>
            <date>15 August 2020</date>
            <country>South Africa</country>
            <region>National</region>
            <availability><ab>Available at: https://mg.co.za/coronavirus-essentials/2020-08-15
                <ab> All material © Mail &amp;amp; Guardian Online. Material may not be published

        </bibl>

    </sourceDesc>
 </fileDesc>

<encodingDesc>

    <classDecl>
        <taxonomy>
            <category>
                <catDesc>News</catDesc>
                <category>
                    <catDesc>
                    Health
                    </catDesc>
                </category>
            </category>
        </taxonomy>
    </classDecl>
</encodingDesc>
```

Figure 15: Example in Figure 14 being represented in XML

After the corpus had been compiled we ran a python program to get the word counts of each category of data, the breakdown is given in Figure 16.

Data Category	Number of texts	Token count
South African twitter data	3287 tweets	50269
South African Blogs	98 articles	59260
Media statements and advisories	101 statements	49850
Political Speeches	106 speeches	182851
South African fiction	10 Books	313887
News websites	803 articles	398966
Total	4405(1118 excluding tweets)	1 055 083

Figure 16: A breakdown of the token counts and number of texts collected for the categories of the corpus

From the contents of the corpus we see that news websites and Books made up over half of the corpus tokens, this was due to different news websites being easily available on the internet and because of this, we attempted to capture the various writing styles of these news sites, this included news websites that were intended for a province or part of a province to news websites intended to target the whole country. For media statements and political speeches, all data was gathered from the South African government website, because the data was coming essentially from one source we decided to include less data for these categories. For blogs and twitter data, less content was included as these categories included very informal language(such as the use of emojis) and at times did not pay much attention to spelling. For twitter data we see only about 3287 tweets although we limited each province to 1500 raw tweets, the reason for this was because many tweets were found to be retweets and as previously mentioned when attempting to get tweets for some provinces the tweets that were gathered were substantially lower than other more popular provinces.

Sub-category	Number of Texts	Token count
National News	125	86 121
Local News	47	22 629
Sport (News)	182	72659
Health (News)	3	3706
Education (News)	2	1011
Lifestyle and Entertainment (News)	264	132 037
Finance (News)	180	80 803
Travel (Blogs)	34	17 829
Lifestyle (Blogs)	64	41 437
South African fiction (Books)	10	313 887

Figure 17: A breakdown of the token counts and number of texts collected for the sub-categories of the corpus

When looking at the token counts and the number of texts for sub-categories Health and Education we see a very low token and text count, the reason for this is because the Health articles could be included in either National News or Lifestyle this is why we included few articles like this in its separate category. Local news data is also low in comparison to national news this was since there were not as many websites available just for local news, most websites focused on national news with some having sub-sections for local news such as news websites called "The Daily Voice". For Blogs we found that most bloggers had articles in the sub-category "Lifestyle" and only had a few articles under the travel sub-category, this is why we see that "Lifestyle" blog data is more than double of the "Travel" blog data.

7 Further Discussion of key results

At the end of development, we ended up with a corpus comprising of text made up of South African English though we believe all the sub-varieties of the language were not fairly represented. Whilst carrying out the experiment it was noted that not much data was available for the lesser developed provinces of South Africa. This was seen with Blogs, news websites, twitter data, and Books as well. Most data were available for the Gauteng, Western Cape, and KwaZulu-Natal provinces. This meant that sub-varieties of SAE used in regions such as Limpopo, Northwest, Eastern Cape, etc were not as prevalent in the corpus as the more popular regions such as Gauteng or Western Cape. Though we do believe that there was enough data for different South African provinces in the Political Speeches category. We also believe the data gathered for the books category was not ideal for

a modern corpus as the books we gathered were over a 100 years old with most books being from the early 1900s, this meant that the language used in the books were not a good representation of the current state of SAE.

When it came to devising an appropriate metadata scheme for the corpus that would add linguistic context and provenance to the corpus, our results were just as expected, and believe we achieved our aim, as the metadata included enough information about the source of the text, such that it could be easily traced back. The metadata scheme also made it easy to search through the corpus and filter the type of texts from the corpus. Examples of this would be like extracting all news articles that were intended for the Western Cape region or extracting news articles belonging to the sub-category "Sport" or even extracting Blogs that were written by a specific author.

From the corpus we achieved at the end of development we believed we had achieved our aim of developing a comprehensive corpus for South African English as we managed to include data from various sub-categories where SAE was used. We also achieved our target of having over 1 million tokens for the corpus.

8 Conclusions

From the results, we saw at the end of the development of the corpus we believe the project was a success as the main aim of developing an electronic South African corpus had been achieved and we had also included a variety of categories where South African English was used.

We do believe though, that a fair representation of all the sub-varieties of South African English was not achieved as previously mentioned, this was due to the lack of availability for data for lesser developed provinces of South Africa.
A comprehensive metadata scheme was achieved as well enabling linguists to have enough background on the source and type of texts that were included in the corpus. This metadata scheme also made it possible to filter through the contents of the corpus by using the constituents of the metadata such as category, author, sub-category, etc.

We believe future works to develop an SAE corpus should implement Part-of-Speech tagging as well, which classifies the type each word is and this could be used to find the type to token ratio of the data and would further aid with linguistic research with respect to an SAE corpus. We also believe more categories of data may be included such as interview transcripts, Political debates, and private conversations as well as implement other techniques for gathering data such as search engine queries, RSS feeds, etc.

References

[1] Gareth Terence Bryant Dwyer. 2014. "Towards Automated Creation and Management of a South African English Web Corpus

[2] Baroni,M and Ueyama,M. 2006 ."Building general- and special-purpose corpora by Web crawling.

[3] Pienaar,L and Klerk, V. (2009). Towards a corpus of South African english: corralling the sub-varieties. Lexikos. 19. 10.4314/lex.v19i1.49135.

[4] Klerk, V. (2002). Starting with Xhosa English … towards a spoken corpus. International Journal of Corpus Linguistics. 7. 21-42. 10.1075/ijcl.7.1.02dek.

[5] Jeffery, C (2003) On compiling a corpus of South African English, Southern African Linguistics and Applied Language Studies, 21:4, 341-344, DOI: 10.2989/16073610309486353

[6] Baroni, Marco, Bernardini, Silvia,Ferraresi, Adriano, Zanchetta, Eros. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. Language resources and evaluation, 43(3), 209–226.

[7] Wynne,M. 2005. Developing Linguistic Corpora: a Guide to Good Practice Chapter 3: Metadata for corpus work (Lou Burnard,University of Oxford © Lou Burnard 2004) pp 40-57

[8] Zhao, L and Kong, W and Wang, Q and Song, L. (2019). Construction of power industry corpus based on data mining and machine learning intelligent algorithm. Journal of Physics: Conference Series. 1187. 022018. 10.1088/1742-6596/1187/2/022018.

[9] Spoustová, J and Spousta, M. (2012). A High-Quality Web Corpus of Czech.

[10] Botha, G and Barnard, E. (2005). Two approaches to gathering text corpora from the World Wide Web.

[11] Spousta, M and Marek, M and Pecina, P. (2008). Victor: the Web-Page Cleaning Tool. 12-17.

[12] Baroni, M and Kilgarriff, A. (2006). Large Linguistically-Processed Web Corpora for Multiple Languages.. Proceedings of EACL 2006. 10.3115/1608974.1608976.

[13] Chrome Web Store. 2020. SelectorGadget. Retreived from https://chrome.google.com/webstore/detail/selectorgadget/mhjhnkcfbdhnjicl

[14] Lexical Analysis Software and Oxford University Press. 2020. WordSmithTools. Retreived from https://lexically.net/wordsmith

[15] Laurence Anthony. 2020. AntConc. Retreived from https://antconc.en.lo4d.com/windows