Computer Science Honours

Final Paper

2016

Title: A spellchecker for isiXhosa

Author: Nthabiseng Mashiane

Project Abbreviation: ALSPEL

Supervisor(s): Dr Maria Keet

Category	Min	Max	Chosen
Requirement Analysis and Design	0	20	15
Theoretical Analysis	0	25	
Experiment Design and Execution	0	20	10
System Development and Implementation	0	15	15
Results, Findings and Conclusion	10	20	10
Aim Formulation and Background Work	10	15	10
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**	**80**

Data-driven statistical model based error detector for IsiXhosa

Nthabiseng Mashiane
Computer Science Honors
mshnth009@myuct.ac.za

ABSTRACT

The proliferation of digitization in our lives warrants the development of tools that ensure efficient digitization and correct usage of data, in this case, textual data. In addition, good quality texts provide accessibility to knowledge and content which could promote further content generation. Word processors ensure that text adheres to the rules of language through validating the data which is presented to it, as per the language rules. Many Bantu languages aren't supported on such tools thus, content creation and access in Bantu languages is scarce.

Very few spellcheckers exist for Bantu languages. There are only two spellcheckers that exist for Bantu languages developed by Ndaba et al.[1] [13] and spellchecker.net. Spellcheckers are used in many computer applications such as word processors, emails and cellphones [13] therefore, it is important that different languages are supported to ensure efficient and correct digitization of data as well as access to correct, readable data which is free of errors.. This paper looks at the development process of a statistical-based model isiXhosa spellchecker, the software development of the spellchecker as well as the outcome thereof. The spellchecker was developed, tested and refined with an accuracy of 79%.

Keywords

Digitization, spellchecking, Corpus, Isolated word error-detection

1. INTRODUCTION

Currently, not many African languages are supported on existing word processors. IsiZulu, IsiXhosa, Sepedi, SeSotho, Setswana, TshiVenda, XiTsonga, IsiNdebele and IsiSwati are collectively known as Bantu languages and are the largest language group in South Africa. Within these Bantu languages exists a sub-group known as Nguni wherein the language being focused on in this paper -isiXhosa - is found. According to the 2011 census of South Africa, approximately 8.1 million people speak isiXhosa which accounts for 16% of the population of South Africa, second to 23% of isiZulu speakers in South Africa. To this end, an isiXhosa spellchecker was created in the course of the project using a data-driven statistical language model in an attempt to create more online tools to support African languages.

A spellchecker is software that analyses possible misspelling in text [17]. Spellchecking mainly comprises of error detection and error correction. Spelling errors can be categorised into two, real-word errors and non-word errors. A real-word is a word which follows the orthographic rules of a language but contains mistakes while a non-word is defined as a set of consecutive letters which does not follow orthographic rules of a language [16]. Spellcheckers are available for languages which carry commercial value such as English, French, Spanish, etc. and less so for some indigenous languages particularly in Africa [16].

Bantu languages are widely spoken yet do not have many tools which aid textual digitization of the language. In addition, these languages are not largely documented online. IsiXhosa is one of the eleven official languages in South Africa that can be used in formal communication including court trials, healthcare access and news content generation. The increase of technological devices to capture such communication is on the rise and can benefit from a spellchecker to improve accuracy when transmitting and translating information. The lack of online tool support in Bantu languages impedes opportunities for research and for non-English speakers to contribute knowledge on a bigger platform such as the internet. Lastly, spellcheckers can aid with the preservation of the language as the data stored can be used at a later stage showing the history of the language and how it has evolved.

The aim of the project is to create a spellchecker for isiXhosa that can correctly perform isolated non-word error detection as there is currently no standalone spellchecker for isiXhosa. A secondary aim of this project is to investigate the accuracy of the isiXhosa spellchecker and assess whether it can achieve the same accuracy or exceed that of the standalone isiZulu spellchecker created by Ndaba et al. [13] which has an accuracy of 89%.

2. LITERATURE REVIEW

This section looks at the related work done on spellcheckers for African languages.

2.1 Digitization of African languages

In South Africa, after democracy, there has been an increase in use of the eleven official languages in official documents as policies have been amended by the state to allow citizens the option of receiving information in their language of choice as opposed to the pre-democratic standard of English/Afrikaans. "Digitization has been defined as the conversion of analogue

[1] https://keet.wordpress.com/2016/11/11/launch-of-the-isizulu-spellchecker/

media to digital form"[1]. It is necessary to digitize African languages as foreign concepts are often imposed on Africa and overwhelm and overpower or heritage [1]. Gibbon et al.[6] further stresses the pertinence of digitization of endangered languages. Digitization of African languages allows for the preservation of the heritage and culture and gives emergence of potential areas of research which could possibly increase the number of linguistic experts particularly in South Africa and other African countries. There is a general insufficient digitization of African languages, but there has been an increasing presence in local languages on the web through channels such as blogs and online publishing forums[18]. Bosch et al.[3] note that there are no standards for digitization let alone machine readable lexica which impedes the digitization of these languages.

Bernstein et al. [2] created a web based interactive word processing interface which enables an online community to aid other members with various writing tasks such as editing, proofreading, formatting, etc. called Soylent. In addition, Soylent enables parties to condense text to meet the required word count in the event that the word count is above the limit as well as a proofreading mechanism written using machine learning algorithms. Moving forward, if a South African spellchecker can take this approach for data collection, it could possibly dissolve the lack of linguistic experts for indigenous languages through the community of native speakers that would engage online. This will allow both expansion of the knowledge and peer review base of the data posted. Some issues noted with this approach are that more often than not, reviewers of work submitted can either do the bare minimum or go above and beyond their requirements. In both cases, extra work is created for the end user [2] which is undesirable for a spell checking tool.

Although texts are now written and are available in various languages in South Africa, not much of these texts are digitized. This is due to the scarcity of linguistic experts in South Africa as well the lack of a standard procedure of data digitization which ensures that the data is captured in a machine readable form [3]. Standards of data digitization could possibly aid in the creation and advancement of new and existing spell checking tools. Lastly, the tools in existence are costly and proprietary e.g. Spelling Checkers for South African languages, WordPerfect 9. There are no effective open source tools in existence and limited funding is available for these tools to be created, thus, the isiXhosa spellchecker developed here will be open source allowing everyone access to it.

2.2 Spellchecking techniques

Spellchecking is made up of two main techniques which are error correction and error detection. Error correction in text has been mainly focused on three areas; non-word error correction, isolated-word error correction and context-dependent word correction while error detection in text has focused on non-word error detection and isolated-word error detection [11]. Error detection involves the analysis of pre-generated n-grams from some language corpus as one of the techniques, these n-grams may be static or dynamic. Error detection has been successfully implemented while error correction is still progressively being worked on. Kukich [11] classifies errors into two; typographical errors which are misspellings and cognitive errors which are errors made by people who don't know how to spell the words. Cognitive errors include phonetic errors as well as errors associated with homonyms that can produce a valid word which is erroneous in context. These two types of errors have to be considered when creating a spell checking tool.

Error correction for South African/Bantu languages is still being developed, with efficient algorithms and tools yet to be found. There exist many other techniques of error detection in addition to the one mentioned above. These include minimum edit distance where the algorithm looks for the smallest number of insertions/deletions to correct a word. Another method is the similarity key technique where strings which have a similar spelling are mapped to identical or the same key so that the key of the misspelled word is similar to that of a correctly spelled word or at least gives possible options of the correct word. This seems like a good approach as Damerau [5] found that 80% of errors in text are a combination of insertion, deletion, substitution and transposition

The structure of African languages is very different from that of the languages currently catered for in spell checking software [7], and this is why not much can be leveraged from the existing spell checkers. String matching algorithms as well as dictionary lookup approaches have been used for spellchecking in Bangla. Neither the string matching algorithms nor the dictionary lookup methods cater for homonym errors i.e. the errors detected aren't specific to context. Moreover, 80% of errors in text are a combination of insertion, deletion, substitution [20, 5].

2.3 Spellchecking advancements

This section looks at the spellcheckers which are currently in existence.

The first spell checker for South African languages was created by D.J Prinsloo [16] in the 1990s. This spellchecker initially worked for isiZulu, Sesotho sa Leboa and Setswana. Later in 2003, Prinsloo [16] improved the functionality of the spellchecker by increasing the size of the wordlists used for spell checking.

Development of spellchecking tools in South Africa has mainly been focused on non-word error detection as opposed to error correction. According to De Schryver et al. [4], non-word error detection works best when tested against Sesotho sa Leboa vs isiZulu and Afrikaans. Binary n-grams have been used successfully for OCR applications, for spellchecking, probabilistic n-grams are used instead. The higher the order of the n-gram tree, the richer the information [4]. Ndaba et al. [13] created an isiZulu standalone spellchecker which performs non-word error detection. This spellchecker was created using the data-driven statistical approach and using character trigrams.

More advanced developments lean towards a more dynamic or automatic error detection where, instead of statically spell checking text after it has been written, spellcheck while they are being typed [9]. Another development is the notion of identifying different languages in texts in order to spellcheck accordingly. Some of the languages in South Africa usually have diacritics which alter meaning of the word and pose issues when the words are encoded and decoded, more so when the encoding mechanism is not specified and because there is no current standard of encoding. This issue is usually encountered when the sources of data are varied (Internet, blogs, etc.) as it makes is difficult to encode and decode characters.

UzZaman and Khan [20] created a Bangla spell checker using generic algorithms which were not tailored to the Bangla language which is spoken in India. In the process, they noted that the orthographic rules are complex and are one of the many reasons why it is difficult to develop efficient algorithms that work for the spell checking tool. Bangla, also known as Bengali, is a language spoken in Southern Asia by approximately 210 million people and is one of the most spoken languages in the world [20].

An Arabic spell checker was created which recognizes common spelling errors and offers suggestions. It was implemented in SICStus Prolog on IBM. The main features of the Arabic language that had to be taken into consideration were computational morphology- which deals with how to derive a new word from an existing one by adding a prefix, suffix or infix and can either change the word category or leave it unchanged. This is referred to as 'morpho graphemic rules' where a word is changed by changing morphological rules. In addition, Arabic has weak and Hamza characters which are characters that are changed by the diacritic of the word. The spell checker has limited its detection to nonwords. After a word is found, various approaches are used to correct the error. Shaalan et al.[19] summarizes the main five spelling errors in Arabic as follows:

1.Reading errors that occur where an individual is capturing data which is written on paper and misreads some of the data, thus capturing the wrong data. In addition to misreading the data, errors arise from lack of certain characters on the keyboard.

2.Another common error is through transcription where the transcriber hears a different thing from what is being said as there are slight nuances in most of the pronunciation of words which mean very different things. Other reasons for these kinds of errors include the presence of various dialects, the use of slang as well as age.

3.Touch typing errors which would usually be from typists who aren't very experienced. And this would be due to the positioning of the typist on the keyboard.

4.Morphological errors which would arise from a writer who doesn't have much experience.

5.Editing errors which are due to typing errors i.e. insertion, delete, subs.

This brief look at the spellcheckers in existence can give rise to potential hybrid approaches to spellchecking and also give evidence to the extent to which statistical and non-statistical approaches have been successful. Overall, it is important to be aware of the different techniques/approaches used for spellchecking as different approaches tackle different aspects of spellchecking and being aware of these aspects will allow one to create a robust spellchecker.

3. SYSTEM DEVELOPMENT AND IMPLEMENTATION

This section discusses the software development process followed while creating the spellchecker, the algorithm and corpus used by the tool as well as the user interface design.

3.1 Software development

A waterfall software development methodology was followed. This methodology was mainly chosen based on the lack of flexibility in the client's schedule. Had an iterative approach been chosen, the client would probably have not been available to evaluate each iteration, thus requirements were gathered at the beginning of the development cycle. The spellchecker was created and tested thereafter.

An initial meeting was held with the client where the goal, requirements and other specifications of the spellchecker were discussed. It was established in that meeting that the client would like to have a Google chrome plugin as opposed to a desktop application. In addition,the client requested that the spellchecker have automatic error detection. Thus, the in the requirements and analysis phase, we agreed that the goal/scope of the project would be a Google Chrome plugin which performs isolated non-word error detection and does automatic error detection.

In the second phase of the development lifecycle, the interface of the Google Chrome plugin was designed and evaluated by Dr Maria Keet, as the client was unavailable. Figure 1 shows the initial design of the plugin and Figure 2 shows the second design after receiving feedback from the initial design. In this phase, the feasibility of the implementation of a chrome plugin was also investigated. The Java backend proved to be difficult to integrate with the chrome libraries and API. Due to time constraints, a desktop application for Windows was created instead. The interface design of the desktop application is discussed in section 3.4

In the third phase of the project, an error detection module was created. This module detects erroneous words in isiXhosa and the error detection is not affected by punctuation.

In the last phase, the tool was tested. A module for testing the accuracy of the spellchecker was created and this is discussed in section 4.1. In addition, the HCI component and usability of the tool were tested, more details are provided about this in section 3.5.

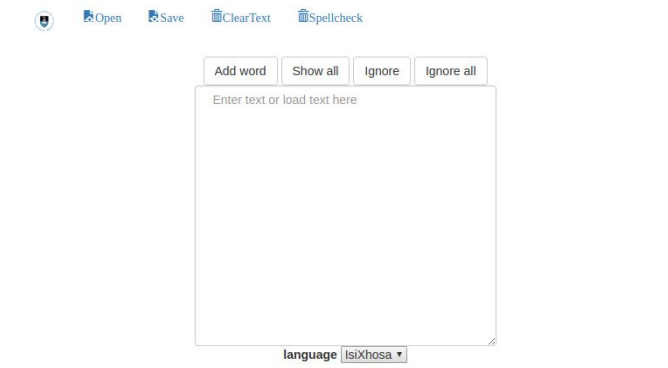

Figure 1 : Initial plugin design

Below is the refined design. It has two text areas, one is for user input and the other displays errors. This is done in real-time as the user types the text. In addition, the user can decide to disable the automatic error detection and use the buttons provided to spellcheck text.

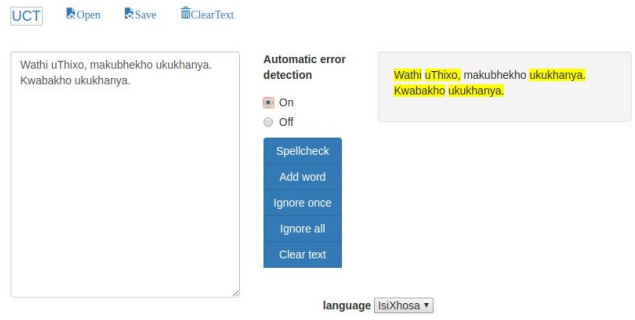

Figure 2: Refined plugin design

3.2 Pre-processing

The isiXhosa documents retrieved from the client, Dr Mantoa-Masoko from the African languages department at the University of Cape Town(UCT) first had to be cleaned and prepared for use by the spellchecker. This step is important as the documents used to create the corpus directly affect the performance of the spellchecker. A cleaner corpus leads to better results. Although no spellchecker is 100% accurate, clean and large corpora improve the performance of the spellchecker.

The data used to build the corpus was cleaned using the method adopted by Norman Pilusa's code for the IsiZulu spellchecker to make the model for the isiXhosa spellchecker. The IsiZulu spellchecker code could only recognize alphabetical characters and not punctuation thus all characters besides alphabetical characters had to be removed. In addition, text written in any other language had to be removed. A module to perform this cleaning was created to read the text and scan each word for characters other than alphabetical characters with the exception of spaces between words. When the module detects non-alphabetical characters, it deletes them.

3.3 Non-word error detection model

Once the data is cleaned, the spellchecker module takes in the words and creates character trigrams. A character trigram is a set of consecutive characters taken from a string e.g. if the string is "molweni" the module would produce "mol", "olw" , "lwe", "wen" and "eni". These trigrams are then stored with their corresponding frequency from the text documents/corpus. The probability of a trigram existing was calculated using the formula below where w_i is the word and N is the total number of words in the corpus.

$$P\ (\ w_i\ /N\ ) = (w_i frequency/\ N\ frequency)$$

The trigram frequency is then compared with a predetermined threshold during error detection. If the frequency of the trigram is below the threshold, the world is flagged as incorrect otherwise, the word is flagged as correct. This means that a bigger corpus gives better results as the error detection module is based on the frequency of the trigrams.

3.4 User interface design

The design process followed was the expert-mindset design where the end user is not a part of the design process but are rather asked to evaluate the design of the software and give feedback and other ideas on how to improve the design.

This project aimed to create a user interface design that was as simple as possible in order to avoid overwhelming and cluttering the user. To evaluate the usability and look of the spellchecker, a usability study was conducted. A group of participants that comprised of students at the University of Cape Town who study isiXhosa at university level or have studied IsiXhosa until grade 12 were asked to partake in the usability testing. The testing session was 30 minutes long. During this time, users were observed as they used the tool to see how quickly they could understand the tasks to be completed and how they proceeded to begin and complete the task. Users were allowed to give feedback in real-time as they used the tool during the study.

The feedback from the participants with regards to the aesthetics and usability of the tool were as follows:

- Users expected the spellcheck button to show all the incorrect words as opposed to showing one incorrect word after the other.
- Users wanted automatic error detection
- Users found the ignore once and ignore all buttons confusing thus tool-tips were added to the buttons
- Users found it hard to open files from the machine.

From this feedback, the spellchecker was refined and now shows all errors within the click of a button, it has tool tips which explain the functionality of the spellchecker and has minimized user clicks to access functionalities.

Figure 3 shows the initial design of the first spellchecker while Figure 4 shows the refined design after user evaluation and feedback.

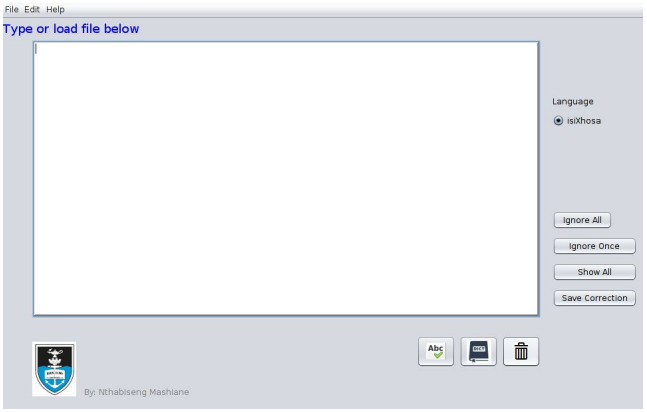

Figure 3: Initial standalone application design

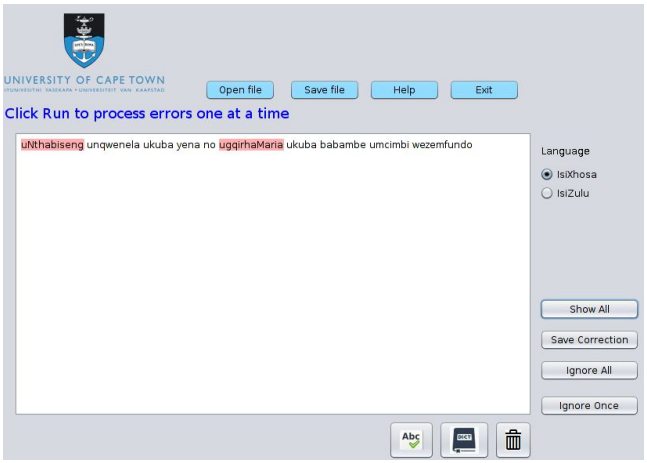

Figure 4: Refined standalone application design

3.5 Testing

The usability of the spellchecker was tested using a usability study comprising of 9 students. Participants were asked to complete tasks using the features on the spellchecker. Once these tasks have been completed, they were asked to answer a set of questions. These tasks and questions were made available using Google forms which allowed the participants to remain anonymous.The set of tasks and questions asked during the study can be found in Appendix A.

The tasks chosen made it possible to evaluate the users' understood the interface design and the components thereof as participants were observed while using the tool and were also able to ask questions during the study if they experienced any difficulty. Only 1 person out of the 9 participants said that they found it difficult to use the tool. Overall, the participants were comfortable with the tool and said that should they write isiXhosa text(s), they would use the tool.

4. EXPERIMENT DESIGN AND EXECUTION

This section describes entities which were altered when creating the spellchecker in order to determine entities that work best.

4.1 N-grams

Character trigrams, quad-grams and quin-grams were created from the word corpus received from Dr Mantoa-Masoko (the client).A trigram is a set of 3 consecutive characters, a quadgram is a set of 4 consecutive characters and a quingram is a set of 5 consecutive characters. For example, if the input word is "Molweni", for trigrams the output would be "Mol", "olw", "lwe", "wen", "eni"; for quad-grams "Molw", "olwen", "lwen" and for quin-grams "Molwe", "olwen", "lweni". In addition to storing these tree structures , the frequency of each n-gram was stored as a key-value pair. The threshold was also altered for each of these tree structures respectively to see which tree structure would produce the highest accuracy. Table 1 shows the tree structure, its threshold and the spellchecker's accuracy.

Tree structure	Threshold	Accuracy
Trigrams	0.002	79.4
	0.003	79.3
	0.004	79.3
Quin-grams	0.002	79.2
	0.003	79.2
	0.004	79.2
Quad-grams	0.002	79.2
	0.003	79.2
	0.004	79.2

Table 1: Accuracy Testing

5. RESULTS AND FINDINGS

5.1 Results analysis and Discussion

This research aimed to investigate the accuracy of the an error detector created using n-grams in comparison to an isiZulu error detector created using the same methodology. The error detector for isiXhosa achieved an accuracy of 79% indicating that a different methodology should be used.

The usability study was conducted with 9 participants. The intended number of participants was 15-20 as this would have allowed for more feedback from participants. There seemed to be miscommunication and misunderstandings when conducting the study as most users had to ask questions to clarify the instructions. This showed that the study needs to be as unambiguous as possible to avoid gathering biased feedback from the participants. Furthermore, an iterative approach with the users as opposed to the client could have been a more viable approach. The users are

generally almost always available thus the design could have been much better.

The statistical approach for the isiXhosa spellchecker performs poorly in comparison to the isiZulu spellchecker. This approach was chosen as it was used by Ndaba et al. [13] for the isiZulu spellchecker and performed very well giving an accuracy of 89%. With isiXhosa and isiZulu belonging to the same language group, the Nguni group, and have similarities in their language structure, it was expected that the statistical approach would work well for both languages but that is not the case. Perhaps a larger corpus would have produced better results

6. CONCLUSION

Bantu languages generally lack online tool support which in turn perpetuates the scarcity of content and the digitization of text in Bantu languages online. The aim of this project was to create a data-driven statistical based model to perform isolated non-word error detection for an isiXhosa spellchecker. This tool was successfully created as a desktop application and achieves an accuracy of 79%. A secondary aim of this project was to investigate whether the isiXhosa spellchecker can achieve an accuracy similar to that of the isiZulu spellchecker. The isiXhosa spellchecker gave a lower accuracy or 79% in comparison with the isiZulu spellchecker which has an accuracy of 89%.The results recorded here are preliminary and can be used as a guide for further development of spellcheckers for Bantu languages.

7. FUTURE WORK

Automatic error detection would be a good feature to add to the spellchecker as it would reduce the number of buttons the user would have to click in order to spellcheck. Also, a Google Chrome plugin could be created using a different language such as Ruby or Python as they have good library support and would possibly be easier to integrate with the Chrome libraries. The spellchecker does not provide error correction as the scope was limited to isolated non-word error detection.

8. ACKNOWLEDGMENTS

A special thank you to my supervisor, Dr Maria Keet for giving me support and advice throughout this project, my second reader, Prof. Sonia Berman for her critique and advice on this project, My client, Dr Mantoa-Masoko for providing the necessary documents for the spellchecker and the participants for their evaluation and feedback during my usability study.

9. REFERENCES

[1] Akinde, T.A., 2007. Digitizing Africa local content: The way forward. ***Continental Journal of Information Technology***,***1***, pp.44-50.

[2] Bernstein, M.S., Little, G., Miller, R.C., Hartmann, B., Ackerman, M.S., Karger, D.R., Crowell, D. and Panovich, K., 2015. Soylent: a word processor with a crowd inside. Communications of the ACM, 58(8), pp.85-94. DOI=10.1145/2791285

[3] Bosch, S.E., Pretorius, L. and Jones, J., 2007. Towards machine-readable lexicons for South African Bantu languages. ***Nordic Journal of African Studies*** **16(2) 131-145**

[4] De Schryver, G.M. and Prinsloo, D.J., 2004. Spellcheckers for the South African languages, Part 1: The status quo and options for improvement. ***South African Journal of African Languages***,***24***(1), pp.57-82.

[5] F.J. Damerau, "A technique for computer detection and correction of spelling errors", communication of ACM, 7(3), 171-176, 1964.

[6] Gibbon, D., Bow, C., Bird, S. and Hughes, B., 2004. Securing Interpretability: The Case of Ega Language Documentation. In LREC.

[7] Grover, A. S., Calteaux, K., van Huyssteen, G., & Pretorius, M. (2010, October). An overview of HLTs for South African Bantu languages. In Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists (pp. 370-375). ACM.

[8] Jackie Jones , Kholisa Podile & Martin Puttkammer (2005) Challenges relating to standardization in the development of an isiXhosa spelling checker, South African Journal of African Languages, 25:1, 1-10 DOI= http://dx.doi.org/10.1080/02572117.2005.10587244

[9] Martin, J.H. and Jurafsky, D., 2000. Speech and language processing. ***International Edition***,***710***.

[10] Maniacky, J. 2003. Umqageli (Automatic identification of Bantu languages). http://www.bantu-languages. com/en/tools/identification.php (Last accessed: 26 April 2003).

[11] Karen Kukich. 1992. Techniques for automatically correcting words in text. ACM Comput. Surv. 24, 4 (December 1992), 377-439. DOI=http://dx.doi.org/10.1145/146370.146380

[12] Keet C.M.,Khumalo,L On the verbalization patterns of part-whole relations in isiZulu. ***Proceedings of the 9th International Natural Language Generation conference 2016 (INLG'16)***,Edinburgh, Scotland, Sept 2016. ACL, 174-183

[13] Ndaba B., Suleman, H., Keet, C.M. and Khumalo, L., 2016, May. The Effects of a Corpus on isiZulu Spellcheckers based on N-grams. In IST-Africa Week Conference, 2016 (pp. 1-10). IEEE. DOI: 10.1109/ISTAFRICA.2016.7530643

[14] Nkomo, D., 2015. Developing a dictionary culture through integrated dictionary pedagogy in the outer texts of South African school dictionaries: the case of Oxford Bilingual School Dictionary: IsiXhosa and English. Lexicography, 2(1), pp.71-99.

[15] Pienaar, W. and Snyman, D.P., 2011. Spelling checker-based language identification for the eleven official South African languages. In Proceedings of the 21st Annual Symposium of Pattern Recognition of SA, Stellenbosch, South Africa (pp. 213-216).

[16] Prinsloo D.J. and de Schryver, G.M., 2003. Non-word error detection incurrent South African spellcheckers. ***Southern***

African Linguistics and Applied Language Studies,***21***(4), pp.307-326.

[17] Randhawa, E. S. K., & Saroa, E. C. S. (2014). Study Of Spell Checking Techniques And Available Spell Checkers In Regional Languages: A Survey. International Journal For Technological Research In Engineering, 2(3).

[18] Scannell, K.P., 2011. Statistical unicodification of African languages. Language resources and evaluation, 45(3), p.375.

[19] Shaalan, K., Allam, A. and Gomah, A., 2003. Towards automatic spell checking for Arabic. In Proceedings of the 4th Conference on Language Engineering, Egyptian Society of Language Engineering (ELSE), Cairo, Egypt(pp. 21-22).

[20] UzZaman, N. and Khan, M., 2006. ***A comprehensive Bangla spelling checker***.BRAC University.

Appendix A

Tasks

1. Enter a full sentence in isiXhosa of your choice and check if it has any errors using the tool.
2. Enter 2-4 sentences in isiXhosa and check them for errors. Ignore all the errors flagged afterwards, if any
3. Open a word or text document from your machine and spellcheck it then clear the contents of the textbox after. The document can be found on file-> desktop->spellcheck -> XhosaText3
4. Enter a paragraph in isiXhosa and ignore an error
5. Open the help functionality. Do you feel as though it explains how to use the spellchecker well enough? Give a brief comment on your answer

Questions

1. How easily were you able to complete the tasks in section 1?
2. Was the tool intuitive?
3. If you answered no to the question above, please state what you was confusing or not intuitive about the tool
4. How complex did you find the system
5. Which features would you like to be added to the spellchecker and why?
6. Which features would like to be removed from the spellchecker and why?
7. Would you use this spellchecker frequently? Please comment on your answer
8. Did the spellchecker meet your needs as a user?Please explain your answer briefly.
9. The spellchecker is approximately 85% accurate. Do you think that is satisfactory or should it be higher?
10. How likely are you to use this tool when writing isiXhosa text? Please give a brief explanation.
11. Do you think a Google chrome plugin would be a better tool to use than a standalone desktop application? Why?
12. Please add any further comments with regards to the accuracy and look of the spellchecker if you have any