TECHLIMED system description for the Shared Task on Automatic Arabic Error Correction

Djamel MOSTEFA
Techlimed
42 rue de l'Université
Lyon, France

Omar ASBAYOU
Techlimed
42 rue de l'Université
Lyon, France

Ramzi ABBES
Techlimed
42 rue de l'Université
Lyon, France

{firstname.lastname}@techlimed.com

Abstract

This article is a system description paper and reports on the participation of Techlimed in the "QALB-2014 shared task" on evaluation of automatic arabic error correction systems organized in conjunction with the EMNLP 2014 Workshop on Arabic Natural Language Processing. Correcting automatically texts in Arabic is a challenging task due to the complexity and rich morphology of the Arabic language and the lack of appropriate resources, (e.g. publicly available corpora and tools). To develop our systems, we considered several approaches from rule based systems to statistical methods. Our results on the development set show that the statistical system outperforms the lexicon driven approach with a precision of 71%, a recall of 50% and a F-measure of 59%.

1 Introduction

Automatic error correction is an important task in Natural Language Processing (NLP). It can be used in a wide range of applications such as word processing tools (e.g. Microsoft Office, Openoffice, ...), machine translation, information retrieval, optical character recognition ... Automatic error correction tools on Arabic are underperforming in comparison with other languages like English or French. This can be explained by the lack of appropriate resources (e.g. publicly available corpora and tools) and the complexity of the Arabic language. Arabic is a challenging language for any NLP tool for many reasons. Arabic has a rich and complex morphology compared to other latin languages. Short vowels are missing in the texts but are mandatory from a grammatical point of view. Moreover they are needed to disambiguate between several possibilities of words. Arabic is a rich language.There are many synonyms and Arabic is a highly agglutinative, inflectional and derivational language and uses clitics (proclitics and enclitics). Arabic has many varieties. Modern Standard Arabic includes the way Arabic is written in the news or in formal speech. Classical Arabic refers to religious and classical texts. Dialectal Arabic has no standard rules for orthography and is based on the pronunciation. Therefore a same word can be written using many different surface forms depending on the dialectal origin of the writer. Another very popular way of writing Arabic on the Internet and the social media like Facebook or Tweeter is to use "Arabizi", a latinized form of writing Arabic using latin letters and digits (Aboelezz, 2009).

For our participation in this evaluation task, we tried to implement two different approaches. The first approach is a lexicon driven spell checker. For this, we have plan to adapt and test state-of-the-art spell checkers. The second approach is a pure statistical approach by considering the correction problem as a statical machine translation task.

The paper is organized as follows: section 2 gives an overview of the automatic error correction evaluation task and resources provided by the organizers; section 3 describes the systems we have developed for the evaluations; and finally in section 4 we discuss the results and draw some conclusion.

2 Task description and language resources

The aim of the QALB Shared Task on Automatic Arabic Error Correction (Mohit, 2014) is to evaluate automatic text correction systems for the Arabic language. The objective of the task is to correct automatically texts in Arabic provided by the organizers. The QALB corpus is used for the evaluation task. A training set and a development set with gold standard is provided for system train-

Proceedings of the EMNLP 2014 Workshop on Arabic Natural Langauge Processing (ANLP), pages 155–159,
October 25, 2014, Doha, Qatar.

ing and development. The training and development sets are made of sentences with errors coming from newspapers articles and the gold standard is made of manual annotations of the sentences. The annotations were made by human annotators who used a correction guidelines described in (Zaghouani, 2014). The corrections are made of substitutions, insertions, deletions, splits, merges, moves of words and punctuation marks.

The training set is made of 19,411 sentences and 1M tokens. The development set includes 1,017 sentences for around 53k tokens.

The evaluation is performed by comparing the gold standard with the hypothesis using the Levenshtein edit distance (Levenshtein, 1966) and the implementation of the M2 scorer (Dahlmeier, 2012). Then for each sentence the Precision, Recall and F-measure are calculated.

Finally a test set of 968 sentences for 52k tokens with no gold standard has to be corrected automatically for the evaluation.

3 System description

For our participation in this evaluation campaign, we studied two main approaches. The first one is a lexical driven approach using dictionaries to correct the errors. Different lexicons were evaluated using Hunspell as spellchecking and correction tool.
The second approach is a statistical machine translation point of view by considering the automatic error correction problem as a translation task. For this we used the statistical machine translation system Moses (Koehn, 2007), to train a model on the training data provided by the organizers.

3.1 Baseline system

Since this the time first we are trying to develop a spellchecker and correction tool for Arabic, we wanted to have some figures about the performance of spellcheckers on Arabic.

We used the development set to test the performance of various spellchecker and correction tools. We corrected the development set automatically using the spellchecker module of the following softwares:

- Microsoft Word 2013
- OpenOffice 2014
- Hunspell

For Microsoft Word and OpenOffice we used the default configuration for correcting Arabic text and disabled the grammar correction.

Hunspell is an open source spellchecker widely used in the open source community. It is the spellchecker of many well-known applications such as OpenOffice, LibreOffice, Firefox, Thunderbird, Chrome, etc. It is the next generation of lexical based spellcheckers in line with Myspell, Ispell and Aspell. It is highly configurable, supports Unicode and rich morphology languages like Arabic or Hungarian. Hunspell uses mainly two files for spellchecking and correction. The first one is a dictionary file *.dic which contains basically a wordlist and for each word, a list of applicable rules that can be applied to the word. The second one is an affix file *.aff which contains a list of possible affixes and the rules of application. More information on these files can be found in the Hunspell manual[1].

Hunspell is an interactive spellchecker. It takes as an input a text to be corrected and for each word that is not found using the loaded dictionary and affix files, it gives a list of suggestions to correct the word. For the correction which must be fully automatic, we forced Hunspell to always correct the word with the first suggestion without any human intervention.

The dictionaries/affixes used for the evaluation is coming from the Ayaspell project(Ayaspell, 2008). The dictionary contains 52 725 entries and the affix file contains 11859 rules.

The results are given in Table 1

Dictionary	Precision	Recall	F-measure
Word	45.7	16.6	24.3
Hunspell	51.8	18.8	27.6
OpenOffice	56.1	20.7	30.2

Table 1: Results on the development set for Word, Hunspell/Ayaspell and OpenOffice(in percentage)

The best results are the ones obtained by OpenOffice with a precision of 56.1%, a recall of 20.7% and a F-measure of 30.2%.

We would like to mention that these spellcheckers do not correct the punctuations which may explain the relative low recall scores.

[1] http://sourceforge.net/projects/hunspell/files/Hunspell/Documentation/

3.2 Statistical machine translation system

Our second approach is to consider the automatic correction problem as a translation problem by considering the sentences to be corrected as a source language and the correct sentences as a target language. Since the organizers provided us with a 1 million tokens corpora with and without spelling errors, we tried to build a statistical machine translation system using the parallel data. We used the Moses (Koehn, 2007), a Statistical Machine Translation (SMT) system to train a phrase based translation model with the training data. The training data provided is made of erroneous sentences and for each sentence a list of corrections to be applied. To build the parallel error/correct text corpus we applied the corrections to the sentences. We came up with a parallel corpus of 19421 sentences and 102k tokens for the error version and 112k tokens for the corrected version. Moses requires a parallel corpus to train a translation model, a development set to tune the translation model and also a monolingual language model in the target language. Since we had to evaluate the performance on the development data provided by the organizers, we had to use part of the training data as a development data for Moses. So we split the 20k sentences included in the training data in a new training set of 18k and a new development data of 2k sentences. We trained standard phrase based models using the surface word form with no morphological analysis or segmentation. For the word alignment in the training process, we used GIZA++ (Och, 2003). The 2k sentences were used to tune the SMT models.

Corpus	# Sentences	Usage
train18k	18000	train
dev-train2k	1411	dev
dev	1017	test

Table 2: Bitexts used for the SMT system

For the language models we used corpora of newspapers publicly available or collected by Techlimed. The sources are coming from the Open Source Arabic Corpora (Saad, 2010) (20M words), the Adjir corpus (Adjir, 2005) (147M words) and other corpora we collected from various online newspapers for a total of 300M words. The language model was created with the IRSTLM toolkit (Federico, 2008).

We evaluated the translation models on the development set using different sizes of monolingual corpus. The 3 systems were trained on the same parallel corpus but with different size for fir monolingual data for System100, System200 and System300 with respectively 100M words, 200M words and 300M words. The results are given in table 3.

System	Precision	Recall	F-measure
System100	70.7	48.8	57.8
System200	70.7	49.6	58.3
System300	70.8	50.1	58.7

Table 3: Results on the development set (in percentage) for the 3 SMT systems

We can see from table 3 that the size of the language model has no impact on the precision but increases slightly the recall of 1.3% in absolute (2.6% in relative).

The BLEU scores (Papineni, 2002) measured on Sytem100, System200, System300 are respectively 65.45, 65.82 and 65.98.

We also tried to combine Hunspell/Ayaspell with the SMT system by correcting the output of the SMT system with Hunspell/Ayaspell but didn't get any improvement.

4 Discussion

The results obtained by the SMT system is much more better than the ones obtained with Hunspell/Ayaspell with a F-measure of 58.7% for the best SMT system and 27,6 for Hunspell/Ayaspell. We have to mention that the training corpus provided by the organizers of 1 million words with the manual annotations enabled us to train a statistical system that learn automatically the correction made by the annotators while Hunspell/Ayaspell was not adapted to the correction guidelines. In particular the punctuations are not corrected by Hunspell/Ayaspell and this explains the difference of recall between the SMT system (50.1%) and Hunspell/Ayaspell (20.7%). If we have a look at the gold standard of the development set, 38.6% of the manual annotations concern punctuation marks with 6266 punctuation marks annotations for an overall total of 16,231 annotations. While there are clear rules for strong punctuation marks like period, question or exclamation marks, there are no clear grammatical rules for the weak punctuation marks, especially for commas which con-

cern 4,117 annotations of the gold standard of the development set (25.4%). Another point that we would like to mention is that a spell checker and correction tool is usually used in an interactive mode by proposing n-best candidates for the correction of a word. When looking at Hunspell/Ayspell correction candidates for an error, we saw the correction was not in position 1 but in the list of candidates. So it would be interesting to compare the correction on the n-best candidates and not only on the first candidate for Hunspell and the SMT system.

5 Conclusion

This paper has reported on the participation of Techlimed in the QALB Shared Task on Automatic Arabic Error Correction. This is the first time we tried to develop a spellchecker for Arabic and have investigated two approaches. The first one is a lexicon driven approach using Hunspell as a spellchecker and correction tool and the second one is a SMT systems using Moses for training a statistical machine translation model on the 1 million tokens corpus provided by the organizers. The best results were obtained with the SMT system which, especially, was able to deal with the punctuation marks corrections. We also tested an hybrid system by combining Hunspell and the SMT system but didn't get better results than the SMT system alone. Our perspective is to improve the results by using hybrid systems based on the DiiNAR lexical database (Abbes, 2004) and also a large arabic named entity dictionary, both owned and developped by Techlimed We will also try to used factored translation models with the Techlimed Part-Of-Speech taggers. And more training data will also improve the quality of the corrections.

Acknowledgments

We would like to thank the QALB Shared Task organizers for setting up this evaluation campaign on automatic error correction tool for Arabic and for providing us with the language resources and tools that we used for the development of our systems.

References

Ramzi Abbès, Joseph Dichy, and Mohamed Hassoun. 2004. The architecture of a standard arabic lexical database: some figures, ratios and categories from the Diinar. 1 source program. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*, pages 15–22. Association for Computational Linguistics, 2004.

Mariam Aboelezz. 2009. Latinised arabic and connections to bilingual ability. In *Papers from the Lancaster University Postgraduate Conference in Linguistics and Language Teaching*, 2009.

Ahmed Abdelali. 2005. http://aracorpus.e3rab.com/

Ayaspell Arabic dictionary project, 2008. http://ayaspell.sourceforge.net

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572. Association for Computational Linguistics, 2012.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. Irstlm: an open source toolkit for handling large scale language models. In *Interspeech*, pages 1618–1621, 2008.

Hunspell, 2007. http://hunspell.sourceforge.net/

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics, 2007.

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

Behrang Mohit, Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, and Ossama Obeid. 2014. The First QALB Shared Task on Automatic Text Correction for Arabic. In *Proceedings of EMNLP Workshop on Arabic Natural Language Processing*, Doha, Qatar, October 2014.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. 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.

Motaz K Saad and Wesam Ashour. 2010 Osac: Open source arabic corpora. In *6th ArchEng Int. Symposiums, EEECS*, volume 10, 2010.

Wajdi Zaghouani, Behrang Mohit, Nizar Habash, Ossama Obeid, Nadi Tomeh, Alla Rozovskaya, Noura Farra, Sarah Alkuhlani, and Kemal Oflazer. 2014. Large scale arabic error annotation: Guidelines and framework. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).