Statistical Denormalization for Arabic Text

Mohammed Moussa, Mohamed Waleed Fakhr
College of Computing and Information Technology,
Arab Academy for Science and Technology,
Heliopolis, Cairo, Egypt
mohammed.moussa@live.com,
waleedf@aast.edu

Kareem Darwish
Qatar Computing Research Institute,
Qatar Foundation, Doha, Qatar
kdarwish@qf.org.qa

Abstract

In this paper, we focus on a sub-problem of Arabic text error correction, namely Arabic Text Denormalization. Text Denormalization is considered an important post-processing step when performing machine translation into Arabic. We examine different approaches for denormalization via the use of language modeling, stemming, and sequence labeling. We show the effectiveness of different approaches and how they can be combined to attain better results. We perform intrinsic evaluation as well as extrinsic evaluation in the context of machine translation.

1 Introduction

Arabic Text Denormalization (ATD) is considered a sub-problem of Automated Text Error Correction (TEC), which is an important topic in Natural Language Processing (NLP). TEC can be used in many applications such as OCR error correction, query-spelling correction, or as pre- or post- processing for other NLP tasks, such as Machine Translation (MT). For example, the training of an MT system that translates between Arabic and other languages is typically improved by normalizing some of the Arabic letters that replace each other depending on context or are commonly confused by document authors. When translating into Arabic, these letter normalizations need to be de-normalized to recover the proper forms of the letters. We test ATD in the context of a:

1. Standalone system for correcting common Arabic mistakes, which are made by users who are not linguistically proficient or who write casually, e.g. bloggers, tweeters, etc. It is also helpful for users who need an automatic system to help them identify such spelling mistakes.
2. Denormalizing MT output when translating into Arabic. We evaluated English to Arabic MT output against properly spelled output and we achieved a BLEU score of 8.78. Upon inspecting the output we saw that normalization during training contributed the most to the low score.

The specific letter normalizations that we will address in ATD are as follows:

1. Restoring ‘ة p’ or ‘ه h’ from ‘ه h’ at the end of a word[1]:

Normalized	Denormalized
سنه snh	سنة snp (year); سنه snh (his age)
هذه h*h	هذه h*h (this fm.)

2. Restoring ‘ى Y’ or ‘ي y’ from ‘ي y’ at the end of a word:

Normalized	Denormalized
وحدى wHdY	وحدي wHdy (alone 1st person)
قصوى qSwY	قصوى qSwY (maximum)

3. Restoring ‘إ <’, ‘أ >’, ‘آ |’ or ‘ا A’ from ‘ا A’:

Normalized	Denormalized
اسلام AslAm	إسلام <slam (Islam)
ارض ArD	أرض >rD (land)
ات At	آت |t (coming)
قال qAl	قال qAl (he said)

These constitute normalizations that are commonly performed for machine translation and information retrieval. Another less common normalization entails conflating ‘ؤ &’, ‘ء '’, and ‘ئ }’ (Darwish and Ali, 2012).

The problem is that a normalized form of an Arabic word may be denomalized into multiple different forms depending on context. For example, both ‘قرآن qr|n’ (Qur’an) and ‘قران qrAn’ (marriage) are normalized to ‘قران qrAn’.

We used two main approaches to solve the problem, namely: using language modeling at word and stem levels; and using Conditional Random Fields (CRF) sequence labeling to handle

[1] We use Buckwalter transliteration throughout the paper

cases not disambiguated using language modeling. We also examined the use of a CRF labeler in isolation of language modeling for comparison.

The main contributions of the paper are:

1. Using stemming in conjunction with language modeling to improve language modeling coverage.
2. Treating the ATD problem as a sequence labeling problem.
3. Using a combined system to achieve a denormalization accuracy greater than 99%.

2 Related Work

Spelling correction is a well-studied problem (Kukich, 1992; Manning and Schutz, 1999). The problem of detecting and correcting misspelled words in text usually involves finding out-of-vocabulary words then finding most similar words in a dictionary using some measure of distance (Levenshtein, 1966; Wagner and Fisher, 1974). Heuristic approaches are also used as in (Shaalan et al., 2003) to find replacement candidates for the misspelled word by adding or removing letters, or splitting words. A finite-state automaton based approach proposed by (Hassan et al., 2008) models letter mapping probabilities and letter and word sequence probabilities.

Though there are commercial systems that perform such denormalization as part of their pipelines, the literature is quite scant on denormalization. (El-Kholy and Habash, 2010) used the MADA analyzer to perform de-tokenization and denormalization.

3 Data and Tools

3.1 Training Data

In our experiments, we used 8 million Arabic sentences, containing 182 million words, from Aljazeera.net news articles to train our language models. Aljazeera.net has very high editorial standards, making spelling mistakes very rare. We constructed language models as follows:

- Word-level models where Arabic text was properly tokenized, and all diacritics (short Arabic vowels), kashidas (word elongations), and numbers were removed.
- Stem-level models where words were stemmed using a statistical stemmer that is akin to AMIRA (Mona Diab, 2009). However we did not remove both 'ة p' and 'ه h' from the end of words as they are letters of interest.

We built our language models using the SRILM toolkit with Good-Turing smoothing (Stolcke, Andreas, 2002).

We trained a character-level CRF model using 5 thousand and 50 thousand sentences. We henceforth refer to this model as the 'CRFModel'. We used the CRF++ implementation (Kudo, 2009) of CRF for all our experiments. The features that we used for the CRF character-level model were as follows:

- Features 1 to 9: Current letter, preceding 4 letters and following 4 letters, each as a feature. This serves as a character language model.
- Features 10 and 11: letter bigram features, namely the current letter with preceding letter, and current letter with following letter.
- Features 11 to 13: letter trigram features, namely current letter with 2 preceding letters, current letter with 1 preceding and 1 following letter, and current letter with 2 following letters
- Features 14 to 17: letter 4-gram features
- Features 18 to 22: letter 5-gram features. Features 10-22 further model Arabic letter sequences, by attempting to capture common 2, 3, 4, or 5 letter sequences that may have consistent denormalization patterns. For example, the sequence 'ال Al' is most likely a determiner that has a single form.
- Features 23 and 24: position of the letter from the beginning and end of a word respectively.

The output labels of the CRFModel are the proper denormalized form of the letter. For example, in case of input 'ا A' the output could be one of the 4 classes 'إ <', 'أ >', 'آ \|' or 'ا A', in case of input 'ي y' the output could be one of the two classes 'ي y' or 'ى Y' and in case of 'ه h' the output could be one of the 2 classes 'ة p' or 'ه h'. The output for the remaining letters would be 'S' (standing for same letter). Thus we had 9 output labels in all.

3.2 Test Data

To test the effectiveness of ATD, we used a test set of three thousand sentences, containing

106,859 words. The test sentences were obtained from islamonline.net, an online site, and were manually checked for errors. Testing involved attempting to perform ATD on a normalized version of the sentences. Around 75% of words required denormalization. Based on the unigram word-level model, 0.8%, 41.8%, 34.3%, 7.5% and 15.6% of the words in the test set have 0, 1, 2, 3 and more than 3 denormalized forms respectively.

To test the effectiveness of ATD in the context of machine translation, we used a parallel English-Arabic test set of 4 thousand sentences containing 36,839 words. We used the Bing online translator to perform MT from English to Arabic. We used BLEU with a single reference translation as the measure of effectiveness. We used two baselines, namely: the output of the Bing system, where the Bing translator performs some sort of ATD, and the same output with an additional letter normalization step. In discussions with the team that worked on the English to Arabic translation in Bing, they indicated that they are using a proprietary denormalization component.

4 ATD Experimental Setups

We used several experimental setups as follows:

1. Unigram LM: In this setup, we simply picked the most common denormalized form of a word regardless of context. If a word is Out Of Vocabulary (OOV), meaning not seen in training, it is left as is. We consider this as our baseline experiment. The approach has the following disadvantages:
   - It ignores contextual ambiguity. For example, though the normalized form علي Ely has the possible denormalized forms: {علي Ely "proper name Ali", على ElY "on"}, the second form will consistently be chosen.
   - Coverage is limited by previously seen words.

2. Unigram Stem LM: Since attached clitics in Arabic typically have 1 form, then the stem (without prefixes and suffixes) is usually the portion of the word that requires denormalization. This setup is identical to the Unigram LM, but denormalization is done at stem level. The advantage of this approach is that it should have better coverage than the Unigram LM. However, it does not take context into account. Also, ambiguity is increased, because attached clitics often disambiguate the correct denormalized form.

3. Unigram LM + Unigram Stem LM: In this setup, we used the Unigram LM setup for all words, and we backed-off to Unigram Stem LM for OOV words. This has the effect of increasing coverage, while using the disambiguation information of clitics. It still ignores context.

4. Bigram LM: In this setup, we generated all known denormalization of a word, and then we used the Viterbi algorithm (bigram model) to ascertain the best denormalized form in context. OOV words were left unchanged. This setup uses context to pick the best denormalization, but it is limited by the previously seen words.

5. Bigram Stem LM: This is identical to Bigram LM, except that the language model is constructed on stems and not words.

6. Bigram LM + Unigram Stem LM: This is identical to Bigram LM, but with back-off to the Unigram Stem LM. This accounts for context and backs-off to better handle OOV words.

7. Bigram LM + Bigram Stem LM: This is identical to Bigram LM, but with back-off to the Bigram Stem LM.

8. CRF Model: We trained the CRF sequence labeler using the aforementioned features. We used the generated CRF model in two ways:
   a. As a back-off to handle OOV words after we apply the entire language model based approaches.
   b. As a standalone approach that attempts to denormalize directly.

5 ATD Experimental Results

5.1 Intrinsic ATD Evaluation

Table 1 reports on the results of using the different language modeling based approaches. Table 2 reports CRFModel as a standalone approach with two different data sizes. Table 3 reports on the same approaches but with CRF-based back-off for OOV words.

Not surprisingly, the results show that using a bigram language model generally produces slightly better accuracy than using a unigram model. Using a stem language model helps only when being used as back-off and that appears clearly for words with more than 1 candidate. This can be explained by the fact that attached clitics can help disambiguate the correct denormalized form. (Results in Table 1 show that 73.7% of OOV's were unchanged after proper denormalization).

Also not surprisingly Table 2 shows that using more data to train CRFModel leads to better accuracy. We chose to use the better CRFModel combined with the language models to report the results in Table 3. Stem and CRF models help in handling OOV's. Table 4 and 5 shows how well these models perform on the words that are left over from word and stem-level models. The CRF model was effective in guessing the proper denormalization for more than 87% of the words.

Candidates/word in LM	All	0	1	> 1
% of test data	100	0.8	41.8	57.4
Setup	Accuracy (%)			
1. Unigram LM	98.2	73.7	99.9	97.3
2. Unigram Stem LM	97.6	86.7	99.6	96.3
3. Unigram LM + Unigram Stem LM	98.3	86.7	99.9	97.3
4. Bigram LM	98.9	73.7	99.9	98.4
5. Bigram Stem LM	98.6	86.4	99.7	97.9
6. Bigram LM + Unigram Stem LM	**99.0**	**86.7**	**99.9**	**98.4**
7. Bigram LM + Bigram Stem LM	99.0	86.4	99.9	98.4

Table 1: Results of using language modeling for ATD

Candidates/word in LM		All	0	1	> 1
% of test data		100	0.8	41.8	57.4
Setup	Data	Accuracy (%)			
8. CRF standalone	5k	95.1	87.5	97.0	93.9
	50k	**97.0** +1.9	**87.7** +0.2	**98.2** +1.2	**96.2** +2.3

Table 2: CRFModel w/ training sets of different sizes.

5.2 ATD Results in MT

Table 6 reports on the BLEU scores for translating 4 thousand sentences from English to Arabic and then performing denormalization using the different approaches. Table 6 reports on two baselines. The first involves not using denormalization at all and the other relies on the denormalization of Bing online translator system. The results show that using our best ATD system edges the Bing system, but the difference is not statistically significant. Using CRF model alone yields results that are 0.38 BLEU points lower than the best system. This shows that even a 2% drop in ATD accuracy may noticeably adversely impact translation quality. When comparing with the Bing translation system, which is nearly state-of-the-art, our proposed ATD system is at par with it. Note that the Bing system has an advantage over our proposed system in that the MT system does not have OOVs in the denormalization phase because it only generates Arabic words that appear in training.

Candidates/word in LM	All	0	1	> 1
% of test data	100	0.8	41.8	57.4
Setup	Accuracy (%)			
1. Unigram LM	98.3 +0.1	88.7 +15.0	99.9	97.3
2. Unigram Stem LM	97.7 +0.1	93.0 +6.3	99.6	96.3
3. Unigram LM + Unigram Stem LM	98.4 +0.1	93.0 +6.3	99.9	97.3
4. Bigram LM	99.0 +0.1	88.7 +15.0	99.9	98.4
5. Bigram Stem LM	98.6 0.0	92.7 +6.3	99.7	97.9
6. Bigram LM + Unigram Stem LM	**99.0** 0.0	**93.0** +6.3	**99.9**	**98.4**
7. Bigram LM + Bigram Stem LM	99.0 0.0	92.7 +6.3	99.9	98.4

Table 3: Results of using language modeling with CRF back-off with relative change over results in Table 1

Setup	Coverage (%)	Accuracy (%)
Unigram Stem LM	54.6	97.0
Bigram Stem LM		96.7

Table 4: Coverage of stem-based models on OOVs

Setup	Accuracy (%)
Word-Based	88.7
Stem-Based	87.5

Table 5: Accuracy of CRF model on OOVs of word-based models and combined stem-based models

Denormalizer System	BLEU non-CRF	BLEU w/CRF
Without ATD	8.78	
Bing Translator	20.79	
Unigram LM	20.75	20.77
Unigram Stem LM	20.64	20.65
Unigram + Stem Unigram LMs	20.76	20.77
Bigram LM	20.80	**20.82**
Bigram Stem LM	20.76	20.77
Bigram + Stem Unigram LMs	**20.81**	**20.82**
Bigram + Stem Bigram LMs	**20.81**	**20.82**
CRF Standalone	20.44	

Table 6: Results for using ATD in MT

6 Conclusion

In this paper, we presented different approaches for performing automatic denormalization of Arabic text to overcome common spelling mistakes and to recover from the normalization that is typically done while training MT systems that translate into Arabic. The different approaches used word language modeling with back-off to a stem-based language models and a CRF model. We tested the different approaches on naturally occurring Arabic text and we evaluated their effectiveness intrinsically and extrinsically in the context of MT. The best technique according to our experiments is a bigram word-level language model with cascaded back-off to a unigram stem language model and then a CRF model to handle the OOVs.

References

A. El-Kholy and N. Habash. 2010. *Techniques for Arabic morphological detokenization and orthographic denormalization*. In Proceedings of Language Resources and Evaluation Conference (LREC)

A. Hassan, S. Noeman and H. Hassan. 2008. *Language independent text correction using finite state automata*. In Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP).

A. Stolcke. 2002. *SRILM – an extensible language modeling toolkit.* In Proceedings of the International Conference on Spoken Language Processing.

C.D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

K. Darwish and A. Ali. 2012. *Arabic Retrieval Revisited: Morphological Hole Filling*. The Association of Computational Linguistics (ACL)

K. Kukich. 1992. *Techniques for automatically correcting word in text.* ACM Computing Surveys.

K. Shaalan, A. Allam and A. Gomah. 2003. *Towards Automatic Spell Checking for Arabic.* In Proceedings of the 4th Conference on Language Engineering, Egyptian Society of Language Engineering (ELSE).

M. Diab. 2009. *Second generation tools (AMIRA 2.0): Fast and robust tokenization, pos tagging, and base phrase chunking*. In Proceedings of 2nd International Conference on Arabic Language Resources and Tools (MEDAR), Cairo, Egypt

R.A. Wagner and M.J. Fischer. 1974. *The String-to-String Correction Problem*. Journal of the ACM

T. Kudo. 2009. *CRF++: Yet another CRF toolkit*. http://crfpp.sourceforge.net.

V.I. Levenshtein. 1966. *Binary codes capable of correcting deletions, insertions, and reversals*. Soviet Physics Doklady.