An enhanced automatic speech recognition system for Arabic

Mohamed Amine Menacer, Odile Mella, Dominique Fohr, Denis Jouvet, David Langlois, Kamel Smaïli

An enhanced automatic speech recognition system for Arabic

Mohamed Amine Menacer, Odile Mella, Dominique Fohr
Denis Jouvet, David Langlois and **Kamel Smaili**
Loria, Campus Scientifique, BP 239, 54506 Vandoeuvre Lès-Nancy, France
{mohamed-amine.menacer, odile.mella, dominique.fohr
denis.jouvet, david.langlois, kamel.smaili}@loria.fr

Abstract

Automatic speech recognition for Arabic is a very challenging task. Despite all the classical techniques for Automatic Speech Recognition (ASR), which can be efficiently applied to Arabic speech recognition, it is essential to take into consideration the language specificities to improve the system performance. In this article, we focus on Modern Standard Arabic (MSA) speech recognition. We introduce the challenges related to Arabic language, namely the complex morphology nature of the language and the absence of the short vowels in written text, which leads to several potential vowelization for each graphemes, which is often conflicting. We develop an ASR system for MSA by using Kaldi toolkit. Several acoustic and language models are trained. We obtain a Word Error Rate (WER) of 14.42 for the baseline system and 12.2 relative improvement by rescoring the lattice and by rewriting the output with the right ء *hamoza* above or below ا *Alif*.

1 Introduction

The Arabic language is the fifth most widely spoken language in the world with an estimated 295 million native speakers. It is one of the most morphologically complex languages. Due to this, developing an Automatic Speech Recognition (ASR) system for Arabic is a very challenging task.

Arabic language is characterized by the high number of dialects used in daily communications. There is a significant difference between these dialects and the Modern Standard Arabic (MSA), which is used in newspapers and formal communication. In this article, we will describe our ASR system for MSA implemented using Kaldi toolkit.

Kaldi is a state of the art toolkit for speech recognition based on Weighted Finite State Transducers (WFST) (Povey et al., 2011; Mohri et al., 2008). It includes multiple scripts and recipes for most standard techniques. These recipes are available with many speech corpora and they are frequently updated to support the latest techniques like Deep Neural Networks (DNN).

In this work, several state of the art's modeling techniques are tested, namely the GMM-HMM models, the DNN models and various techniques like: Maximum Mutual Information (MMI) (Bahl et al., 1986), feature-space Maximum Likelihood Linear Regression (fMLLR) (Povey and Saon, 2006) and Speaker Adaptive Training (SAT) (Anastasakos et al., 1996). The gain obtained after training each model will be reported later on.

Our ASR system is built using several hours of standard Arabic news broadcasts from corpora distributed by ELRA.

Another interesting treatment, proposed in this article, is the auto-correction of ء *hamoza* in the ASR system output in order to rectify the orthography confusion of this symbol above or below ا *Alif*. The approach used is inspired from various techniques proposed in the literature for detection and correction of spelling errors. The particularity of our approach is the use of the vector representation of words to retrieve the context and to correct misspelled words.

In the next section, an overview about Arabic language issues and some works proposed in the literature to deal with those problems is presented. Section 3 describes the different corpus used to train the acoustic and language models, as well as the data normalization process. Section 4 details the acoustic and language models. Finally, the ex-

perimental results are discussed in Section 5.

2 Related works

Even though classic techniques for ASR systems can be efficiently applied to Arabic speech recognition, it is necessary to take into account language specificities to improve the system performance. Arabic is a morphologically rich language. By concatenating prefixes and suffixes to stems, other words are obtained. The stem can be also decomposed into a root (generally a sequence of three consonants) and a pattern of vowels and, possibly, additional consonants. For example: the word وبكتبهم *wabikutubihim* "*and with their books*" is composed of the two prefixes و *w* "*and*" and ب *b* "*with*", the stem كتب *kutub* "*books*", which is derived from the root كتب *ktb* "*to write*" and the suffixe هم *hum* "*their*". This explains the high out-of-vocabulary (OOV) rate compared with English language which consequently leads to the increase of the Word Error Rate (WER). To deal with this issue, (Afify et al., 2006; Xiang et al., 2006; Diehl et al., 2009; Ng et al., 2009) propose to use morphological segmentation. They shown that the results obtained with a large lexicon could be achieved with a reduced one if morphological decomposition is applied.

Another interesting approach investigates language models based on morphological analysis. Choueiter et al. (2006) used a morpheme-based language modeling by exploiting a statistical segmentation algorithm (Lee et al., 2003) to decompose data. An automaton of type finite state acceptor was used to allow legal sequences of morphemes. With this approach, a 2.4% absolute WER improvement was achieved by using a medium vocabulary (less than 64k words) and a morpheme n-gram model compared to a conventional word-based model. However, by using a large vocabulary (800k words), an absolute improvement of only 0.2% was achieved.

Likewise, the Factored Language Models (FLMs) (Bilmes and Kirchhoff, 2003) was used to improve the WER. In (El-Desoky et al., 2010), the morphological decomposition was combined with FLM to iron out the Arabic complex morphology. A good improvement was shown by rescoring the n-best list with a FLM based on partially decomposed words.

One more idiosyncrasy of the Arabic language is that it is a consonantal language. It just has three vowels, each of which has a long and short form. Formal texts are generally written without short vowels, consequently a single grapheme word could have several possible pronunciations. For example, the word كتب *ktb* could be pronounced like: كَتَبَ *kataba* "*write*", كُتُبٌ *kutubN* "*books*" or كُتِبَ *Kutiba* "*written by*" and it also has other potential diacritizations. This ambiguity is solved by using the contextual information of words. Even though the short vowels make easy the pronunciation modeling, their use increases the number of the entries in the vocabulary and consequently the size of the language model. In fact, El-Desoky et al. (2009) showed that the best WER value is achieved by applying a morphological decomposition on a non-diacritized vocabulary. However, a nice improvement was shown in (Kirchhoff and others, 2002) by using short vowels in data training transcripts.

Besides short vowels, another problem to be taken into account in pronunciation modeling is the geminated consonants. In fact, there are cases where the consonant pronunciation should be stressed, and this can frequently happen with the prefix ال *Al* "*the*". The solar consonants after this prefix should be doubled (the solar consonants are: ت *t*, ث *v*, د *d*, غ *g*, ر *r*, ز *z*, س *s*, ش *$*, ص *S*, ض *D*, ط *T*, ظ *Z*, ل *l*, ن *n*). The matter of geminated consonants was investigated in some studies. In (Lamel et al., 2009), it has been shown that modeling explicitly geminates improved a little bit the system performance.

Another issue in Arabic concerns the omission of the symbol ء *hamoza* which is pronounced but often not written. This leads to a pronunciation ambiguity. For example: the word العب *AlEb* could be pronounced ألعب *>aloEab* "*I play*" if the *hamoza* is above *Alif* أ or إلعب *<ilEab* "*play*" if it is below *Alif* إ.

3 Data resources

The data presented in this section are utilized to train acoustic and language models, to estimate the different parameters and to test the performance of the system.

3.1 Acoustic data

To train the acoustic model, a collection of spoken transcribed data-set is required. In our case, we used two corpora: Nemlar[1] and NetDC[2] distributed by ELRA. They consist of several hours of Standard Arabic news broadcasts recorded in linear PCM format, 16 kHz and 16 bits.

The data was splitted into three parts: one part for training (Train), the second for tuning (Dev) and the last one for evaluating the performance of our system (Test). Table 1 illustrates some statistics about the acoustic data.

Corpus	Train	Dev	Test	Total
Nemlar	33(83%)	3(08%)	3(9%)	40
NetDC	19(82%)	3(10%)	2(8%)	23
Total	52(83%)	6(09%)	5(8%)	63

Table 1: The acoustic data (hours).

The data splitting is done randomly by keeping 52 hours for the Train, which is equivalent to 83% of data. 6 hours (9% of data) are used in the Dev set and the rest (5 hours) is used in the Test set. In order to balance the data selection between the two corpora, two-thirds of the data is selected from Nemlar corpus.

3.2 Textual data

The language model is trained by using two corpora: GigaWord[3] Arabic corpus and the acoustic training data transcription.

GigaWord corpus was collected from nine sources of information with a total of 1,000 million word occurrences. The transcription of the acoustic training data contains about 315k words.

As regards the lexicons, the Nemlar and NetDC corpora are provided with phonetic lexicons in Arabic SAMPA format. We used them in the training task in order to specify the pronunciation of each word in the transcription of acoustic training data. The two lexicons have 79k pronunciation variants and 77k unique vowelized words, which is equivalent to an average of 1.02 pronunciation variants per word. The number of pronunciation variants per word is weak because all data transcripts and lexicons are written with short vowels and thus each word will not have various pronunciation.

In the recognition task, we used non-diacritized data for training language model, therefore another lexicon without short vowels is used. This lexicon will be described in Section 4.2.

3.3 Data normalization

Several issues were encountered while processing the textual corpora due to the Arabic spelling, which is often ambiguous. Therefore, a normalization step is necessary when processing the Arabic text.

Most of the orthographical errors were treated by using regular expression rules. In following, some processing necessary for reducing the ambiguity of spelling and pronunciation are presented:

- All email addresses, url paths, special characters (<, $amp; ...), punctuations and non-Arabic texts are removed.
- All diacritics representing short vowels or consonant stressing are striped.
- All numbers are normalized and they are converted into literal words.
- The prefix و *wa* "*and*" is separated from words by using Farasa toolkit (Abdelali et al., 2016) and all other prefixes: ب *b*, ف *f*, ال *Al*, ك *k*, ل *l* and س *s* are concatenated to words.
- The stretched words are reduced to their original form. For example: الرجاااال is replaced by الرجال "*men*".
- A space is inserted after all words end by a ة *ta marobuTa* if it is attached to the next word. For example: replace نهايةالقرن by نهاية القرن *nihAyat Aloqaron* "*century end*".
- The time is literally written such as in the following example: replace 15:30 by الثالثة و ثلاثون دقيقة *Alv∼livap wa valAvwn daqyqap*.
- Some abbreviations are replaced by their corresponding meaning (see table 2 for some examples).

[1] http://catalog.elra.info/product_info.php?products_id=874

[2] http://catalog.elra.info/product_info.php?products_id=13&language=fr

[3] https://catalog.ldc.upenn.edu/LDC2011T11

Abbre	Word	English gloss
%	في المائة	Percent
ت غ	توقيت غرينتش	GMT
ت	تاريخ	Date
هـ	هجري	Islamic Calendar
د.ك	دولار كندي	canadian dollar
س	ساعة	Hour
د	دقيقة	Minute
ث	ثانية	Second

Table 2: Abbreviations and their corresponding meaning.

4 Modelization

In this section, the different steps involved in the development of the DNN acoustic model is presented. Afterwards, the language modeling aspects and the various models developed are detailed.

4.1 Acoustic model

The development of the acoustic model is based on the Kaldi recipe. Our purpose here is to train a DNN model, which perform well with respect to the WER. For this, six different acoustic modeling systems are developed. For three of them, the emission probability of the HMM states is modeled by Gaussian Mixture Models (GMM) and for the others it is modeled by DNN models.

The acoustic features used are the Mel-Frequency Cesptral Coefficients (MFCC) with first and second order temporal derivatives. Therefore, the feature dimension is 39.

Three GMM-HMM models are successively trained. The first acoustic model (triphone1) is trained using directly the MFCC features. A Linear Discriminative Analysis (LDA) followed by a Maximum Likelihood Linear Transform (MLLT) are applied to train the second acoustic model (triphone2). For the third model (triphone3), the Speaker Adaptive Training (SAT) transformation with feature-space Maximum Likelihood Linear Regression (fMLLR) are used to make the system independent of speakers.

In order to take into account the influence of the context on the acoustic realization of the phones, all these models are triphone based models. The last model (triphone3) has 100k Gaussians for 4,264 states.

The DNN-HMM systems are trained using the frame-level cross entropy, sMBR criterion, the senone generated from the last GMM-HMM model (triphone3) and corresponding fMLLR transforms. In total, three DNN models are trained.

- DNN1 classifies frames into triphone-states, i.e it estimates Probability Density Functions (PDFs). DNN1 training is based on the cross-entropy criterion.
- DNN2 and DNN3 are based on sMBR sequence-discriminative training. The difference between the two models is the number of iterations used to train the model. The sMBR sequence-discriminative training is used to train the neural network to jointly optimize for whole sentences instead of a frame-based criterion.

The DNN models have 6 hidden layers and 2048 nodes per layer. The input layer has 440 nodes (40-dimensional fMLLR features spliced across 5 frames on each side of the central frame) and the output has 4,264 nodes. The number of parameters to estimate is about 30.6 millions.

Figure 1 summarizes all the acoustic models of our ASR system.

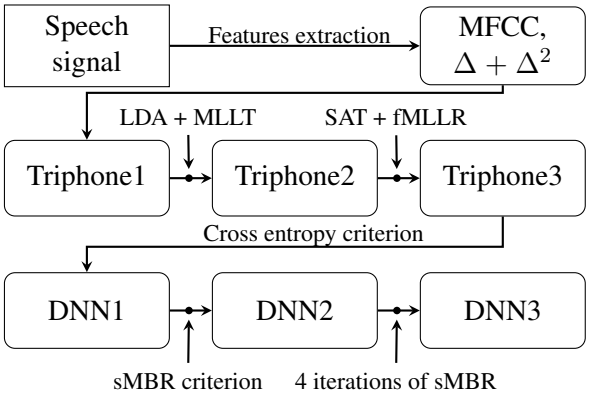

Figure 1: Acoustic model flow diagram.

4.2 Language modeling

A 2-gram Language Model (LM) is used to generate the lattice and a 4-grams LM is used to rescore this lattice. As the two text corpora available (GigaWord and transcripts of Train corpus) are unbalanced, a conventional training process is used. A LM is first trained on each data set (one on GigaWord and one on transcripts). They are then merged through a linear interpolation, where the

optimal weights are determined in order to maximize the likelihood of the transcripts of the Dev set, which has a size of about 31k words.

For the 4-gram LM, which is used to rescore the lattice, 10 LMs are interpolated. Nine of them are trained on the different sources of GigaWord corpus and the last one is trained on the transcripts of the Train data. The interpolation coefficients are again estimated on the transcripts of the Dev data set.

The recognition vocabulary (lexicon) is generated by first keeping the 109k most frequent words from GigaWord corpus and the words that appear more than 3 times in the transcripts of the Train corpus. Afterwards, only the words for which pronunciation variants are in the Nemlar, NetDC lexicons and the lexicon used in (Ali et al., 2014) were kept. This process has generated a lexicon having 95k unique grapheme words and 485k pronunciation variants, that is an average of 5.07 pronunciation variants per word. The high number of pronunciation variants per word is due to the fact that the lexicon entries do not contain the indication of the short vowels. Hence several pronunciation variants are possible for each word. This lexicon is used as the vocabulary to train the language models.

The SRILM toolkit (Stolcke, 2002) has been used to train the different LMs and all of them use Good-Turing (Katz) (Katz, 1987) smoothing technique. It is known that the Kneser-Kney smoothing (Chen and Goodman, 1996) performs better than the Katz technique. However, in (Chelba et al., 2010), the authors showed through different experimental setup that the Katz smoothing performs much better than the Kneser-Kney smoothing for aggressive pruning regimes, which is the case in our system. In fact, due to memory constraints while compiling the automaton used by Kaldi for speech decoding, we used 2-grams pruned language models to generate the lattice. The pruning is done by keeping the n-grams with probability greater than 10^{-9}. The 4-grams language model has also been pruned according two approaches. The first approach is the same as the one used to prune the 2-grams LM and the second is based on stolcke pruning technique (Stolcke, 2000). This second pruned 4-grams LM is presented in Section 5.2.

The n-gram number and the perplexity calculated on the transcripts of the Dev data for various models before and after pruning are presented in Table 3.

***n*-gram**	**unpruned**	**pruned**	**Stolcke pruning**
1-gram	95 589		
2-grams	69 307k	20 164k	2 449k
3-grams	327 302k	22 283k	1 395k
4-grams	586 722k	4 967k	192k

(a) Number of *n*-grams in the interpolated language models.

***n*-gram**	**Perplexity**
2-gram	246.76
2-grams (pruned)	258.22
4-grams	178.48
4-grams (pruned)	189.45
4-grams (stolcke pruning)	214.58

(b) 2 and 4-grams models perplexity.

Table 3: Statistics about LMs used to generate and to rescore the lattice.

5 Evaluations

This section presents the speech recognition results obtained with a 95k word lexicon for the baseline system, and after rescoring lattices. We also proposed an approach to auto-correct the ء *hamoza* above or below ا *Alif* to improve the performance.

5.1 Baseline system

Speech recognition engines determine the word sequence W which maximises the combination of two scores: the acoustic score $P(O|W)$ and the linguistic one $P(W)$. However, these two scores are calculated on different data which leads to a different scale of probabilities. In fact, the language model score is greater than the one provided by the acoustic model. The probabilities are adjusted as follows:

$$\hat{W} = \arg\max_{W} P(O|W)P(W)^{LM} \qquad (1)$$

where LM is a fudge factor.

In order to estimate the best value of LM, we used the transcripts of the Dev corpus. Figure 2 presents the evolution of the WER with respect to the language model weight LM for each acoustic model.

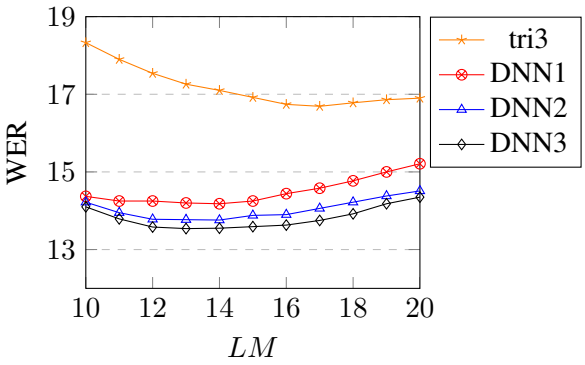

Figure 2: WER evolution with respect to the language model weight.

In table 4, the best values of LM for each acoustic model are presented, as well as the WER calculated on the Dev (31,314 running word) and the Test (31,726 running word) sets. Note that the lattice, in this baseline system, is generated by using the pruned 2-grams language model.

Model	LM_w	**Dev WER**	**Test WER**
tri3	17	16.69	17.65
DNN1	14	14.18	15.23
DNN2	14	13.76	14.61
DNN3	13	13.54	14.42

Table 4: WERs for baseline systems (without rescoring and by using the 2-grams LM).

As expected, DNN models perform better than the GMM-HMM models. The best WER value is 14.42 obtained by using the DNN3 model, which is based on four iterations of sMBR sequence-discriminative training. It should be noted that another GMM-HMM model is trained by applying the Maximum Mutual Information (MMI) criterion. By this, the WER decreased from 17.65 to 16.86 (a relative improvement of 4%). By using the DNN model, a relative reduction in WER of 14.47 has been achieved with respect to GMM-HMM model.

It should be also noted that OOV rate is about 2.35% for the Dev part and 2.54% for the Test.

5.2 Rescoring

Let's recall that Kaldi is based on Weighted Finite State Transducers (WFST) for decoding. Because of this constraint, the decoding is done with a 2-grams LM. One can expect that a rescoring using a more detailed LM (e.g., 4-grams) would improve performance. Thus, we applied a 4-grams rescoring, but only on the DNN3 hypotheses.

WFST is an automaton, which has a set of states and a unique start state. These states are interconnected by arcs, where each arc has an input label, an output label and a weight. To accomplish the language model rescoring, Kaldi generally first subtract the old language model cost from the global score and then add in the new language model cost to avoid modifying the other parts of the global score.

When using this approach, it is more accurate to replace a 4-grams model by another 4-grams LM. For this, we pruned the full 4-grams LM by using stolcke pruning technique (Stolcke, 2000). This technique is based on minimizing the relative entropy between the full and the pruned model. We get a model which represents only 30% of the original model and consisting of 4×10^6 n-grams. It should be noted that the pruning is done by using the pocolm toolkit[4]. We used this new model to produce the lattice. Afterward, this lattice is rescored by using a full 4-grams LM.

As in the baseline system, we estimate the impact of the LM weight on the Dev data. The variation of the LM weight is illustrated in Figure 3. We can remark that the smallest value of WER is obtained for $LM = 14$.

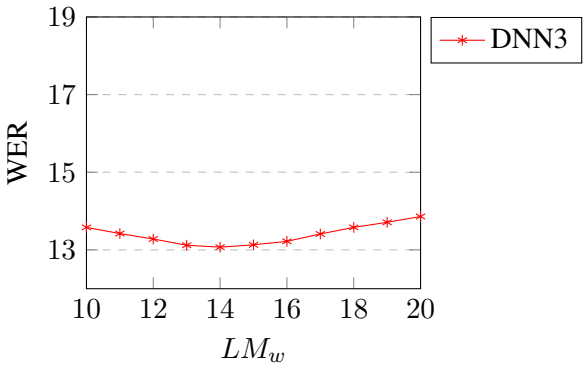

Figure 3: WER with respect to the language model weight after rescoring the lattice.

The evolution of the WER with or without rescoring is given in Table 5.

[4]https://github.com/danpovey/pocolm

Model	Dev WER	Test WER
DNN3	14.65	15.32
DNN3+rescoring	13.07	14.02

Table 5: WERs before and after rescoring the whole lattice produced by using the 4-grams pruned LM.

Rescoring the whole lattice with the 4-gram LM leads to an absolute improvement of 1.58% on the Dev set and 1.3% on Test corpus in comparison to the system, where the lattice is produced using the pruned 4-grams language model (the LM with 4×10^6 n-grams).

We can also remark that producing the lattice by using a 2-grams pruned LM gives better results than using a 4-grams LM pruned with an aggressive pruning regimes. This is justified by the number of n-grams in each model (the number of n-grams in the 2-grams LM is 5 times greater than the number of n-grams in the 4-grams pruned LM).

6 Auto-correction of *hamoza*

The ء *hamoza* symbol is widely used in Arabic; by analyzing the ASR system output, we noticed that there are cases where the symbol ء *hamoza* above or below ا *Alif* is omitted. Therefore, it seems interesting to auto-correct the *hamoza* spelling.

Our approach is inspired from techniques proposed in the literature to detect and auto-correct the spelling errors. This issue is a common problem to all languages. In Arabic, the most frequently occurring errors are editing errors and semantic spelling errors. The first error type occurs when a correctly spelled word is replaced by a non-word, while in the semantic spelling errors, the word is replaced by another correctly spelled word (Alkanhal et al., 2012).

Several works have been proposed for spelling auto-correction in Arabic. Most of these works are based on the three steps described below.

Error detection: Techniques used in the literature for detecting Arabic spelling errors are essentially based on two approaches: the language rules (AlShenaifi et al., 2015; Shaalan et al., 2010; Hassan et al., 2014) or a dictionary (Attia et al., 2014; Zerrouki et al., 2014; Alkanhal et al., 2012). For the first technique, detecting whether a word is misspelled or not depends on morphological analyzers. While the dictionary based technique depends on a large word list that covers the most frequently used words in the language.

The technique which we used to detect *hamoza* error is the dictionary lookup, where the input word is considered such as a non-word if it is not found in the dictionary. The word list size used in our case is 9.2M words. It is developed by Attia et al. (2012) by amalgamating various Arabic resources.

Production of hypotheses: The most common used technique to produce candidates is based on an edition distance, which measures the difference between two sequences by calculating the number of required edits to transform a word into another. We used a modified version of Damerau-Levenshtein distance proposed in (Alkanhal et al., 2012). The idea behind this distance is to assign a low distance cost for the letters that have shape or pronunciation similarity, or keyboard proximity. As we just want to correct *hamoza* error, we considered the similarity between the letters ا, أ, إ and آ. For example: consider the misspelled word امر *Aamara* "*to order*", if we calculate the similarity between the misspelled word and the two correct words تمر *tamor* "*date*" and أمر *>amara* "*to order*" using the normal Damerau-Levenshtein distance, we will obtain the same cost=1. While the modified distance will assign a lower cost for the word أمر *>amara* "*to order*" than تمر *tamor* "*date*" because of the similarity between the two letters أ and ا. At the end, to produce correction hypotheses, we just considered words in the Arabic word list, which have the same spelling as the wrong word except for the *hamoza* above or below ا *Alif* (أ and إ. In the case where any candidature is not found, the word will not be corrected.

Error correction: for error correction i.e. selecting the best solution among the list of candidates, we tried to retrieve the words context by using word2vec (Mikolov et al., 2013). In fact, we used the GigaWord corpus to train a *cbow* model and to obtain word vectors, which are positioned in a 200-dimensional space such that words that share common contexts in the corpus are located in close proximity to one another in the space. Afterwards, we used the cosine similarity to retrieve the most similar word among the candidates.

Table 6 shows the results before and after cor-

recting the *hamoza* in the ASR system output.

Model	Dev WER	Test WER
Baseline	13.54	14.42
Correction	13.03	14.14
Rescoring	13.07	14.02
Rescoring+correction	**12.26**	**13.45**

Table 6: WERs before and after correcting the *hamoza*.

From Table 6, note that the WER for the baseline system is 14.42. It should be noted that the lattice in this system is generated by using the 2-grams LM. By correcting the *hamoza* in the ASR system output, we improved our ASR baseline system by 2%. The WER after rescoring the lattice generated by using the pruned 4-grams LM is 14.02. This represents 3% relative reduction in WER comparing to the baseline system. The best WER is 13.45 obtained by correcting the system output after rescoring. This leads to 7% relative improvement.

7 Conclusion

In this article, we described an ASR system for MSA developed by using Kaldi toolkit. We presented the different acoustic models trained and the text pre-processing done before training the LMs. The best results are achieved by rescoring the lattice, which is generated by using the DNN model, a 4-grams pruned LM and a lexicon of 95k words. This way we have obtained 3% relative improvement. In order to improve the system output, we proposed an approach based on the edit distance to auto-correct the *hamoza* spelling above or below *Alif*. Applying this approach, we achieved an improvement of 12% relative in comparison to the baseline model.

References

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for arabic. In *HLT-NAACL Demos*, pages 11–16. Association for Computational Linguistics, San Diego, California.

Mohamed Afify, Ruhi Sarikaya, Hong-Kwang Jeff Kuo, Laurent Besacier, and Yuqing Gao. 2006. On the use of morphological analysis for dialectal arabic speech recognition. In *Interspeech*.

Ahmed Ali, Yifan Zhang, Patrick Cardinal, Najim Dahak, Stephan Vogel, and James Glass. 2014. A complete kaldi recipe for building arabic speech recognition systems. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 525–529, Dec.

Mohamed I Alkanhal, Mohamed A Al-Badrashiny, Mansour M Alghamdi, and Abdulaziz O Al-Qabbany. 2012. Automatic stochastic arabic spelling correction with emphasis on space insertions and deletions. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(7):2111–2122.

Nouf AlShenaifi, Rehab AlNefie, Maha Al-Yahya, and Hend Al-Khalifa. 2015. Arib@ qalb-2015 shared task: A hybrid cascade model for arabic spelling error detection and correction. In *ANLP Workshop 2015*, page 127.

Tasos Anastasakos, John McDonough, Richard Schwartz, and John Makhoul. 1996. A compact model for speaker-adaptive training. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 2, pages 1137–1140. IEEE.

Mohammed Attia, Pavel Pecina, Younes Samih, Khaled Shaalan, and Josef van Genabith. 2012. Improved spelling error detection and correction for arabic. In *The International Conference on Computational Linguistics (COLING)*, pages 103–112, Mumbai, India, 14 December.

Mohammed Attia, Mohamed Al-Badrashiny, and Mona Diab. 2014. Gwu-hasp: Hybrid arabic spelling and punctuation corrector. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Langauge Processing (ANLP)*, pages 148–154.

LR Bahl, Peter F Brown, Peter V De Souza, and Robert L Mercer. 1986. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *proc. icassp*, volume 86, pages 49–52.

Jeff A Bilmes and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, pages 4–6. Association for Computational Linguistics.

Ciprian Chelba, Thorsten Brants, Will Neveitt, and Peng Xu. 2010. Study on interaction between entropy pruning and kneser-ney smoothing. pages 2422–2425.

Stanley F Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics.

Ghinwa Choueiter, Daniel Povey, Stanley F Chen, and Geoffrey Zweig. 2006. Morpheme-based language modeling for arabic lvcsr. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE.

Frank Diehl, Mark JF Gales, Marcus Tomalin, and Philip C Woodland. 2009. Morphological analysis and decomposition for arabic speech-to-text systems. In *Interspeech*, pages 2675–2678.

Amr El-Desoky, Christian Gollan, David Rybach, Ralf Schlüter, and Hermann Ney. 2009. Investigating the use of morphological decomposition and diacritization for improving arabic lvcsr. In *Interspeech*, pages 2679–2682.

Amr El-Desoky, Ralf Schlüter, and Hermann Ney. 2010. A hybrid morphologically decomposed factored language models for arabic lvcsr. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 701–704. Association for Computational Linguistics.

Youssef Hassan, Mohamed Aly, and Amir Atiya. 2014. Arabic spelling correction using supervised learning. *arXiv preprint arXiv:1409.8309*.

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401.

Katrin Kirchhoff et al. 2002. Novel speech recognition models for arabic. In *Johns-Hopkins University summer research workshop*.

Lori Lamel, Abdelkhalek Messaoudi, and Jean-Luc Gauvain. 2009. Automatic speech-to-text transcription in arabic. *ACM Transactions on Asian Language Information Processing (TALIP)*, 8(4):18.

Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam, and Hany Hassan. 2003. Language model based arabic word segmentation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 399–406. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2008. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*, pages 559–584. Springer.

Tim Ng, Kham Nguyen, Rabih Zbib, and Long Nguyen. 2009. Improved morphological decomposition for arabic broadcast news transcription. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4309–4312. IEEE.

Daniel Povey and George Saon. 2006. Feature and model space speaker adaptation with full covariance gaussians. In *Interspeech*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December. IEEE Catalog No.: CFP11SRW-USB.

Khaled Shaalan, Rana Aref, and Aly Fahmy. 2010. An approach for analyzing and correcting spelling errors for non-native arabic learners. In *Informatics and Systems (INFOS), 2010 The 7th International Conference on*, pages 1–7. IEEE.

Andreas Stolcke. 2000. Entropy-based pruning of backoff language models. *arXiv preprint cs/0006025*.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Interspeech*, volume 2002, page 2002.

Bing Xiang, Kham Nguyen, Long Nguyen, Richard Schwartz, and John Makhoul. 2006. Morphological decomposition for arabic broadcast news transcription. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE.

Taha Zerrouki, Khaled Alhawaity, and Amar Balla. 2014. Autocorrection of arabic common errors for large text corpus. *ANLP 2014*, page 127.