Unsupervised Method for Improving Arabic Speech Recognition Systems

Mohamed Labidi
LaTICE laboratory
Unit of Monastir
5000 Monastir, Tunisia
labidi8mohamed@gmail.com

Mohsen Maraoui
Computational
Mathematics Laboratory,
Tunisia
5000 Monastir, Tunisia
maraoui.mohsen@gmail.com

Mounir Zrigui
LaTICE laboratory
Unit of Monastir
5000 Monastir, Tunisia
mounir.zrigui@fsm.rnu.tn

Abstract

One of the big challenges connected to large vocabulary Arabic speech recognition is the limit of vocabulary, which causes high out-of-vocabulary words. Also, the Arabic language characteristics are another challenge. These challenges negatively affect the performance of the created systems. In this work we try to handle these challenges by proposing a new unsupervised graph-base method. Finally, we have obtained a 4.6% relative reduction in the word error rate. Comparing our suggested method with other methods in the literature, it has given better results. Moreover, it has presented a major step towards solving this problem. In addition, it can be easily adaptable to other languages.

1 Introduction and state of the art

One of the big challenges in speech recognition is how to cover all possible words by a speech recognition system. The vocabulary of a conventional large-vocabulary continuous speech recognition system is finite, and this vocabulary limits the terms that appear in speech transcriptions. The words that do not occur in the vocabulary of the recognizer are called “out-of-vocabulary” words. This problem is a perennial challenge for speech recognition, where the out-of-vocabulary words are badly recognized. A larger vocabulary for the automatic speech recognition system is not the solution, since language is in constant growth and new words are steadily enriching the vocabulary. In (Ng and Zue, 2000), an analysis of news text demonstrated that the vocabulary size would continue to grow as the dataset got larger. In other words, it was not possible to create single large vocabulary that would eliminate the out-of-vocabulary problem. Consequently, it was not possible to create a language model that would cover all the words of any language. Furthermore, under certain conditions, adding more words could compromise the recognition performance of words already in the vocabulary. According to (Logan et al., 2005), up to 10% of all query words in a typical application that used a word-based recognizer with large vocabulary could be out-of-vocabulary words. Of course it was possible to update the vocabulary of the Automatic Speech Recognition (ASR) systems by adding new words to the language model. However, as noted by (Logan et al., 2005), it could be difficult to obtain enough training data to train the language model for new words. Additionally, for most application scenarios, it would not be feasible to re-recognize spoken content once the initial transcription was generated, due to the high computation cost of the ASR process and the huge sizes of daily spoken content collections. For these reasons, the out-of-vocabulary problem was a formidable one.

For the Arabic language, this problem limits the performances of speech recognition systems. As noted in the previous paragraph, it is not practical to recreate a new language model each time we want to enrich our systems by new vocabulary. To deal with these problems, some superficial work has been done. In (Novotney et al., 2011), a morpho-base language model was used in speech recognition systems for four morphologically rich languages which were Turkish, Finnish, colloquial Egyptian Arabic and Estonian. The authors said that the experiments showed that the morph models performed fairly well on out-of-vocabulary words without compromising the recognition accuracy on in-vocabulary ones. Nevertheless, they reported that the Arabic language was the exception where their proposed method failed. They noted that

this might be due to the Arabic language characteristics. The second work belongs to (El-Desoky et al., 2009), where the authors addressed the out-of-vocabulary problem and the non-appearance of diacritical-marks at the Arabic written transcriptions. The authors introduced a morphological decomposition, as well as a diacritization in Arabic language modeling. Their experiments showed a reduction in the Word Error Rate (WER) by 3.7%. However, they still suffer from the new words in languages and diacritical marks in the Arabic words, which present a big problem for Arabic speech recognition. Other work related to this topic has been done in other domains, as in (Al-Shareef and Hain, 2012), (Razmara et al., 2013), (Creutz et al., 2007), (Diehl et al., 2009) and (Habash, 2009).

In our work, we investigate a graph-based method to deal with the present challenge. We use our web crawler to collect text data from the Internet on a regular, continuous and up-to-date basis. We use the collected text for the construction of an oriented weighted graph, where each node presents a word and each arc presents the relationship of succession between two words in the Arabic language. After that, we use a graph search method to detect the false words in the transcription. Finally, we discover the best words that can be replacements.

The paper is organized as follows. In section 2, we present our methodology of performing false-word correction and we deal with out-of-vocabulary words. Our experiments are discussed in section 3, while section 4 gives the conclusions.

2 Methodology

In this section we describe how the corrections of false words are performed. Figure 1 describes the steps of the work.

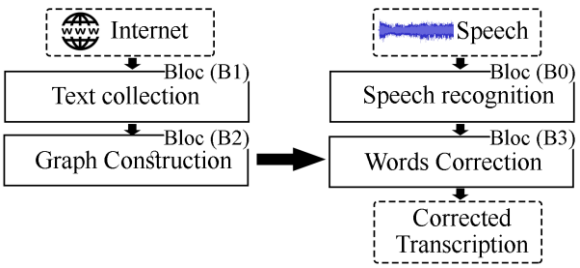


Figure 1: Architecture of the proposed system.

2.1 Linguistic tools

Our acoustic model is built with the help of the CMU Sphinx (Lamere et al., 2003). We train it using 51h of audio material for the modern standard Arabic, recorded by 41 native speakers. Each audio file is accompanied by its transcription. The audio files are converted to 16 kHz, 16 bits, mono speakers, and in an MS WAV format, as required by the Sphinx trainer. The phonetic dictionary is similarly used by almost all researchers in the construction of Arabic speech recognition systems (Ali et al., 2009).

Our language model training corpora consist of around 200 million running full words including data from Ajdir Corpora, Tashkeela corpora (Zerrouki and Balla, 2017), Abbas corpora (Abbas et al., 2011), OSAC corpora (Saad and Ashour, 2010) and collected corpora. Our statistical language model is constructed using the SRILM toolkit (Stolcke and others, 2002).

To evaluate the recognition performance, our small audio corpus of 8h for all our experiments is divided into 12 audio files. Each one contains almost 40 minutes of speech. They contain almost 48,000 Arabic words where 2,000 of them are out of vocabulary (they do not exist in the vocabulary of the system).

For the construction of the oriented weighted graph we use our web crawler to collect text from the Internet and our Java implementation to construct the graph, where each sentence in the collected corpus is transformed to a set of connected words in the graph (i.e., each node of the graph contains one word).

2.2 Speech recognition (B0)

To make the speech correction, it is much easier to work on the text more than spoken documents. For this reason, we have to use a speech recognition system to get the transcriptions of the spoken documents.

We use the CMU Sphinx tools to construct our speech recognition systems. The utilized data are described in the linguistic tools section (section 2.1) and the obtained results are described in section 3. The system gives us the transcriptions for the recognized speech files.

2.3 Text collection (B1)

The text collection is a process to collect Arabic texts from the Internet to establish a corpus of

Arabic text. We use our web crawler in this task. It proceeds as follows:

- Search for the addresses of Arabic web sites in the Internet using API search engines.
- Only Keep addresses of authentic sites: (using the WOT tool, which is a tool powered by 140 million users, machine learning, which is a free browser extensions, and mobile app and API, which let us check whether a website is safe and contains correct information before reaching it).
- Save the authentic addresses in a database.
- Parse the authentic web pages and collect the Arabic texts.
- Save the collected Arabic texts in files (text corpus).

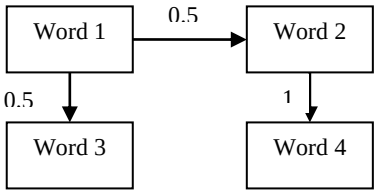
The first successful execution of our web crawler allows collecting more than 2,981 Arabic text files. The advantage of our web crawler is that it systematically updates the corpus. That way we guarantee that our corpus is updated and increased each time. We guarantee also that each new word in the language will be added as soon as possible. The collected corpus is used to create our oriented weighted graph in the next section.

The graph is systematically auto-updated by new texts from the Internet, which make it bigger day after day. The update of the corpus follows the next steps:

- Search for the addresses of Arabic web sites in the Internet using API search engines.
- Only Keep addresses of authentic site: (using the WOT tool).
- For each found authentic address check whether it does not exist in our database, then save it; else do not save it.
- Parse the authentic web pages and collect the Arabic texts.
- Save the collected Arabic texts in files (text corpus).

2.4 Graph construction (B2)

Using the collected corpus in the previous section, where our web crawler is issued, we create an oriented weighted graph that depicts the Arabic language words succession (Figure 2). Each word in the corpus is transformed to a node in the graph. And each two words that succeed in the corpus they will linked by an arc in the graph as described in the following table.



(a) Graph illustration

Node
Word
Number of occurrences
Date of first use
Next nodes
Weight of next relations

(b) Node structure

Figure 2: Illustration of the constructed oriented weighted graph and the structure of its nodes.

The graph of Figure 2 presents the relationship of succession between the four words and the probabilities of these successions. Where the value (0.5) that exist on the arc between “word 1” and “word 2” presents the probability P(“word 2”| “word 1”). It is systematically auto-updated by new texts from the Internet, which make it bigger day after day. This graph is used to correct false words in the transcription.

Each node in the graph is a word from the corpus. Also, it contains only one Arabic word and the information related to it. (a) describes the node structure and its fields. Hence, each sentence in the corpus is transformed to a set of connected nodes in the graph. The following points describe the following node fields.

- Word: Field containing the word
- Number of occurrences: Field containing the number of occurrences of the word in the corpus

- Date of first use: Field containing the first appearance of the word in the Internet or in documents
- Next nodes: Links to the next nodes
- Weight of the next relations: Field containing the weight of the relations between the current word and the next words.

To create our graph we pass by the following steps:

- Create for each word in the corpus a node in the graph. Each word has only one node in the graph, even if it exists several times.
- If a word "X" comes after another word "Y" in the textual corpus, then the node of the word "X" will be linked by an arc to the node of the word "Y" in the graph. The following example explains how two words can be transformed to the graph and how we make the link between them.

In the text	In the graph
« Hello word »	Word X → Word Y

Table 1: Illustration of the arc construction between words.

The arc between any two words "W" and "Y" is weighted by P(W|Y), which is the probability of the appearance of "W" and "Y" together such as that "Y" arrives after "W".

2.5 Word correction (B3)

Our goal in this section is to correct the false words in the transcriptions using the graph created in section 2.4. The correction passes by the steps explained in the next sections:

Suppose we have the following sentence, which contains a false word (Word 3).

Word1 Word2 Word3 Word4 Word5 Word6

To correct the false word (Word 3) we follow the following steps:

2.5.1 False-word detection

First of all, we should detect the false words in the transcriptions, for that we use the oriented weighted graph created in section 2.4. The graph contains the Arabic words collected from the Internet, books, journals, etc. Added to that, the graph is automatically updated by the new words that appear in the language. Logically, any correct word in the transcription should be presented in the graph. To know whether a word is false or not, we search for it in our created graph. If it exists, then it will be correct. Else, it will be considered as a false word and it will pass to the correction step.

2.5.2 Context-window construction

The context window is a set of words that appears with the false word in the same sentence or in the same phrase. It contains N words from both the left and the right of the false word. The context window is used to search correct words that appear in the same context as our false word. Table 2 gives an example of the context-window construction.

Therefore, each false word has more than one context window. Each context window has a different size. The size of the context windows for a false word starts from N=1 (one word from the left and one word from the right of the false words) and reaches N=N, which is the maximum number of words that appear with the false word in the transcription.

We vary the size of the context window for each false word in order to search for the most appropriate context window size that filters out the best possible replacements for the false word. We consider the best context window size as the size that gives us the minimum of possible replacements. We make this choice because we consider that the context window which gives the minimum number of replacements is a better semantic filter than the windows which give more replacements.

The false word (Word 3) in the following example :

Word1 Word2 Word3 Word4 Word5 Word6

has 2 words on the left and 3 words on the right. Two or more of these words can describe the context of the false word that we want to replace. The number of words of the context window (N words) cannot exceed 3 in the example provided in section 2.5, because this is the maximum number of words that can be found with the false

word (Word 3) in one of its two sides (left and right).

Context window size	N=1	N=2	N=3
Left side	Word 2	(Word2-Word1)	(Word2-Word1)
Right side	Word 4	(Word4-Word5)	(Word4-Word5-Word6)

Table 2: Example of context-window construction.

2.5.3 Search for possible replacements

After the construction of the context windows, we search for possible replacements of false words, using the context windows created in the previous section. We search in the graph for the word that has the same context window as our false word. We take the words order of the context windows into consideration. For example, if the false word "word3" appears between the two words "word4" and "word2" in the transcription, then we search in the graph for replacements that appear between "word4" and "word2".

The result of this search step is a set of words. Each set contains a set of possible replacements for the false word. Also, each set presents the search results using one of the context windows of the false word; i.e., for each context window for the false word, this step will give us a set of possible replacements. Table 3 describes the created context windows for the false word (Word 3) given in as example in the following sentence : "Word1 Word2 Word3 Word4 Word5 Word6".

Context window size	N=1	N=2	N=3
Possible founded replacement	Word X Word Y Word Z Word W	Word X Word Z	Word Y Word Z Word W

Table 3: Example of searching possible replacements.

The next section describes the selection of the best set of replacements for the false word.

2.5.4 Selection of best set of replacements

The best context window is the one that gives us the replacements that are semantically the closest to the false word in its context. Then, the best context window will give us the minimum possible of replacements because it filters the words well and it proposes only the semantically closest words to the false one. Thus, the best set of replacements is the one that contains the minimum number of replacements. This step is explained in Table 3 and Table 4.

Context window size	N= 2
Possible replacement	Word X Word Z

Table 4: Example of selecting the best replacement set.

2.5.5 Replacement of false word

In the previous step, we chose the set of replacements that were semantically closest to the false word because they have the same context and it works as a semantic filter. Researchers usually choose one word as a substitute to the wrong one. For us, we opt for replacing the false word by all possible replacements selected from the previous step. On the other hand, each replacement is put with its probability of succession that appears in the graph. This probability defines its relationship of succession of the replacement with its successor and predecessor. This process is explained in the following example.

We suppose that the replacements appear in the graph as represented in Figure 3 where

"Word X" and "Word Z" are the possible correct replacements of the false word.

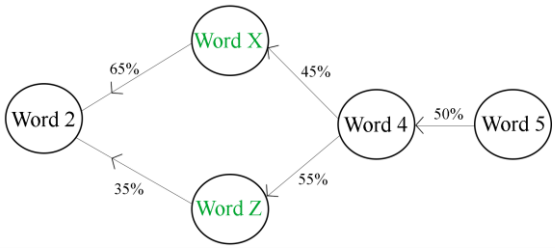


Figure 3: Replacement relations in the graph.

These two possible replacements will replace the false word in the transcription as described in Table 5. Where, the false word is replaced by its possible replacements. And each replacement is accompanied by its probabilities of successions between it and the words of the contextual window.

Replacing false word by the selected ones
Word1 Word2 (65%)WordX (45%) Word4 Word5 Word6
Word1 Word2(35%)WordZ (55%) Word4 Word5 Word6

Table 5: Replacing the false word.

3 Experiments

Our experiments are decomposed in two parts. The first one is the post-correction experiments where we evaluate our speech recognition system performance before the use of our proposed method. The second one is the correction experiments where we evaluate our suggested method. We evaluate our correction method twice: the first one before updating the graph and the second one after updating it. The material used in the experiments is described in the experimental setup section just after the introduction. We use the WER metric, because it is mostly used by researchers to evaluate automatic speech recognition systems (Ali et al., 2009), (Diehl et al., 2009).

3.1 Experiments results

WER% before correction	12.5%
WER% after first correction	8.11%
WER% after second correction (after updating the graph)	7.9%

Table 6: Tests results.

Table 6 shows the obtained results. The first line describes the WER obtained with our speech recognition system before the correction step. The obtained WER is 12.5% ,which means that the transcription contains 6,000 wrongly recognized words, including the 2,000 out-of-vocabulary words. After that, to decrease the WER we execute our proposed method. The second line of Table 6 contains the WER% obtained after the execution of our correction approach, which is 8.11%. This execution was released with the graph constructed in section 3.3. We notice that the WER is decreased. We have recorded a gain of 4.39% in terms of WER, which means a reduction in the number of the false words. We pass from 6,000 to 3,896 false words in the transcriptions. Then, 2,104 words are corrected and 956 of them are out-of-vocabulary words.

After the correction step, we update our graph automatically. Then, we relaunch the correction again, but this time with a richer graph. Line 3 of Table 6 indicates the obtained results. The WER becomes 7.9%, with a reduction of 0.21% from the previous correction; i.e., we pass from 6,000 false words in the transcription to 3,792 ones. However, the number of the corrected out-of-vocabulary words is bigger this time. We pass from 956 corrected out-of-vocabulary words in the first correction to 1,148 ones in the second correction, which proves that the update of the graph has added new words and has positively influenced the correction process.

Work	**Gain in WER%**
(El-Desoky et al., 2009)	3.7%
Our method	4.6%
(Messaoudi et al., 2006)	1.2%
(Afify et al., 2005)	1.4%

Table 7: Comparison between methods.

The obtained results show the efficiency of our proposed method in the detection and correction of false words. In addition, the results show the ease, speed and performance of our method in the enrichment of the corpus and in correction, unlike the classical language models and the difficulties of their enrichment. As cited in the methodology section, our method does not replace the false word by another word from the

possible replacements, but it replaces it by all possible replacements accompanied by their probabilities, which gives a huge advantage to the transcription so that it can be used in various fields. Moreover, any researcher can utilize any selection method to give preference to the suitable word. Furthermore, Table 8 shows that our proposed method gives better results and deals better with false words and out-of-vocabulary ones in the Arabic speech recognition systems than that of the most recent work in the field.

3.2 Discussion

We have proposed a method to correct badly recognized words by any Arabic speech recognition system. Our method shows a good performance in the correction task. Furthermore, it shows an admirable performance in dealing with out-of-vocabulary words. This is thanks to our proposed graph which is systemically auto-updated by new vocabulary and texts from the Internet. Also, it gives a probabilistic description for the words succession in the language. Our method shows a better correction rate than other methods in the literature (El-Desoky et al., 2009), (Creutz et al., 2007) especially for out-of-vocabulary words. In addition, our proposed method provides better results because it takes into consideration the Arabic language characteristics. All this gives our method a great advantage over other ones. Besides, our proposed method can be adapted to other languages easily.

We believe that the correction of false recognized words in any transcription given by any Arabic automatic speech recognition system should take into account two major points. The first is the language characteristics and the second is the new vocabulary that is appearing in the language day after day. Our proposed method is a good step in this field and it can be improved by other methods like the rule-based ones. This is going to be our goal during the next work.

4 Conclusion

In this paper we have tried to deal with the challenges of the limit of vocabulary and the Arabic language characteristics in large vocabulary Arabic speech recognition systems. We have tested a graph-based method. It has given a good reduction by 4.6% in terms of WER. Furthermore, it has fairly dealt with the Arabic language characteristics. The proposed method presents a good step in this field and in dealing with the challenges. Another important thing is that our method can be easily adapted to work with other languages.

References

Mourad Abbas, Kamel Smaïli, and Daoud Berkani. 2011. Evaluation of Topic Identification Methods on Arabic Corpora. *JDIM*, 9(5):185–192.

Mohamed Afify, Long Nguyen, Bing Xiang, Sherif Abdou, and John Makhoul. 2005. Recent progress in Arabic broadcast news transcription at BBN. In *Interspeech*, volume 5, pages 1637–1640.

Mohamed Ali, Moustafa Elshafei, Mansour Al-Ghamdi, and Husni Al-Muhtaseb. 2009. Arabic phonetic dictionaries for speech recognition. *Journal of Information Technology Research (JITR)*, 2(4):67–80.

Sarah Al-Shareef and Thomas Hain. 2012. CRF-based Diacritisation of Colloquial Arabic for Automatic Speech Recognition. In INTERSPEECH, pages 1824–1827.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. ACM Transactions on Speech and Language Processing (TSLP), 5(1):3.

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.

Nizar Habash. 2009. REMOOV: A tool for online handling of out-of-vocabulary words in machine translation. In Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR), Cairo, Egypt.

Paul Lamere, Philip Kwok, William Walker, Evandro B Gouvêa, Rita Singh, Bhiksha Raj, and Peter Wolf. 2003. Design of the CMU sphinx-4 decoder. In INTERSPEECH. Citeseer.

Beth Logan, J-M Van Thong, and Pedro J Moreno. 2005. Approaches to reduce the effects of OOV queries on indexed spoken audio. IEEE transactions on multimedia, 7(5):899–906.

Abdelkhalek Messaoudi, J Gauvain, and Lori Lamel. 2006. Arabic broadcast news transcription using a one million word vocalized vocabulary. In Acoustics, Speech and Signal Processing, 2006.

ICASSP 2006 Proceedings. 2006 IEEE International Conference on, volume 1, pages I–I. IEEE.

Kenney Ng and Victor W Zue. 2000. Subword-based approaches for spoken document retrieval. Speech Communication, 32(3):157–186.

Scott Novotney, Richard M Schwartz, and Sanjeev Khudanpur. 2011. Unsupervised Arabic Dialect Adaptation with Self-Training. In INTERSPEECH, pages 541–544.

Majid Razmara, Maryam Siahbani, Reza Haffari, and Anoop Sarkar. 2013. Graph Propagation for Paraphrasing Out-of-Vocabulary Words in Statistical Machine Translation. In ACL (1), pages 1105–1115. Citeseer.

Motaz K Saad and Wesam Ashour. 2010. Arabic morphological tools for text mining. Corpora, 18:19.

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

Taha Zerrouki and Amar Balla. 2017. Tashkeela: Novel corpus of Arabic vocalized texts, data for auto-diacritization systems. Data in Brief, 11:147–151.