A Supervised Approach to Arabic Text Summarization Using AdaBoost

Riadh Belkebir and Ahmed Guessoum

Natural Language Processing and Machine Learning Research Group,
Laboratory for Research in Artificial Intelligence, Computer Science Department,
University of Science and Technology Houari Boumediene (USTHB), BP 32 El-Alia,
Bab Ezzouar, 16111 Algiers, Algeria
belkebir.riadh@gmail.com, aguessoum@usthb.dz

Abstract. In recent years, research in text summarization has become very active for many languages. Unfortunately, looking at the effort devoted to Arabic text summarization, we find much fewer attention paid to it. This paper presents a Machine Learning-based approach to Arabic text summarization which uses AdaBoost. This technique is employed to predict whether a new sentence is likely to be included in the summary or not. In order to evaluate the approach, we have used a corpus of Arabic articles. This approach was compared against other Machine Learning approaches and the results obtained show that the approach we suggest using AdaBoost outperforms other existing approaches.

Keywords: Artificial Intelligence, Intelligent Systems, Machine Learning, Arabic Natural Language Processing, Text Summarization, AdaBoost.

1 Introduction

The huge amount of documents coming from internet has fostered the need to develop intelligent systems helping to search, access and read documents quickly. One of the goals of Artificial intelligence is to provide users with intelligent systems that help summarizing texts automatically. Text summarization is seen as one of the most challenging applications in Natural Language Processing (NLP). In recent years, this field has gained a lot of interest from the community. A number of organizations and prestigious conferences have started paying a lot of attention to it. Among these, we can mention the Document Understanding Conference (DUC)[1] which has been organized from 2001 to 2007, the Text Analysis Conference (TAC)[2] which has organized a summarization task from 2008 to 2011 and the Association for the Advancement of Artificial Intelligence (AAAI)[3].

[1] http://www-nlpir.nist.gov/projects/duc/
[2] http://www.nist.gov/tac/
[3] http://www.aaai.org/

Text summarization is either extractive or abstractive. Extractive summarization approaches aim to select the most important content of the original document and concatenating them to generate a summary. Abstractive text summarization on the other hand aims to create a new, shorter document (a summary) from the source document by allowing an internal representation and using advanced natural language generation techniques to generate the summary. In this case, new fragments, which were not necessarily present in the source document could be added to generate the summary. Due to its complexity, this paradigm appears to be the ultimate goal in text summarization [14].

Our work focuses on Arabic text summarization. We mention that Arabic is the official language of 22 countries. According to a recent study[4], Arabic is the fastest-growing language on the web with an annual growth of 5,296.6% in 2013 for the number of internet users. Chinese has an annual growth rate of 1,910.3%, 1,123.3% for Spanish, French with 557.5%, Russian with 2,721.8% and English with 468.8%. There are a total world population for Arabic of about 367 millions. Compared to other languages, research on Arabic Natural Language Processing (ANLP) is still shyish. As a consequence, Text Summarization, a sub-field of ANLP, has also had much fewer attention for Arabic than for other languages.

One way to address the problem of text summarization is by means of machine learning techniques. These techniques have shown their usefulness in various natural language processing applications such as information retrieval, text classification, machine translation and speech recognition. The aim of machine learning techniques is to learn a hypothesis(function) instead of algorithmically programming it. In other words, instead of knowing exactly how to solve a problem, in whichever area, one of the machine learning techniques can be used to learn the way to solve the problem from examples. As such, by feeding pairs of input-output, i.e. of characteristics of the problem such as an original text in our case (input) and a representation of the solution such as the summary of the original text (output), the technique used learns a hypothesis, i.e. how to automatically produce a summary from a text. As we will see later in Section 2, some machine learning techniques have been proposed in the literature to tackle the text summarization problem and have been rather successful. In this paper, we propose a new approach to text summarization based on AdaBoost which we will explain in Section 3.

The main idea of our work is to build a hypothesis that could select the most relevant sentences in the source document so as to keep the most important information as well as preserve the readability of the summary. In our case, the inputs are sentences and the outputs are the corresponding labels. Two types of labels are considered. Label "one" (1) is used to indicate that the sentence should be included in the summary and "zero" (0) to indicate that the sentence should be discarded from the summary. These decisions are taken by the AdaBoost learning model.

[4] `http://www.internetworldstats.com/stats7.htm`

The reminder of this paper is as follows. Section 2 presents related research from the literature. Section 3 describes the design of the approach adopted in this work. Section 4 presents the results of the implementation. A conclusion is then given in Section 5.

2 Related Work

For languages like English, and some other European languages like French, Swedish, and Spanish, various approaches have been developed to deal with text the summarization task [9,12,11]. A lot of effort is also underway for some Asian languages such as Japanese, Chinese and Indian. Unfortunately, research on Arabic text summarization is still very much underdeveloped compared to that on the aforementioned languages. This section reviews the main approaches that have tackled the problem of Arabic text summarization.

In [2] the authors present an automatic extractive Arabic text summarization system. Their system uses Rhetorical Structure Theory (RST) in conjunction with a sentence scoring scheme. Summaries of different lengths were compared to those made by a human expert. They explained that their system overcomes some of the other existing systems including those based on machine learning techniques. In [5] a multi-document summarizer is presented despite their complaint about the lack of Arabic multi-document corpora and gold standard for text summarization. The results produced by this system were compared to five systems in the DUC2002 multi-document summarization task. In [7] a comparative study between three approaches for automatic summarization of Arabic documents is presented. The first method is based on a symbolic approach; the second uses a numerical approach while the third one is a hybrid between the two. They show that the numerical approach outperforms the symbolic one while the hybrid approach outperforms the other two. In [1] probabilistic neural networks (PNN) are used with several features. They studied the effect of each feature on the text summarization task. Then, they used a combination of all the features to train the probabilistic neural network (PNN) so as to generate a text summarizer. Lakhas [4] is a system that generates 10-word summaries from news articles. The first step consists in summarizing the source articles; then it translates the documents into English. This approach was considered very successful for very short (headlines) summaries.

3 AdaBoost-Based Summarizer

3.1 AdaBoost Algorithm

The boosting mechanism was introduced by Schapire [13] for boosting the performance of a weak learning algorithm. After several improvements AdaBoost [6] was presented by Bauer & Kohavi [3]. The general AdaBoost algorithm is as follows.

Input: training set S of size m, Inducer $\mathcal{I}$, integer T (number of trials).

```
1.  $S' = S$ with instance weights assigned to be 1.
2.  For $i = 1$ to $T$ {
3.      $C_i = \mathcal{I}(S')$
4.      $\epsilon_i = \frac{1}{m} \sum_{x_j \in S' : C_i(x_j) \neq y_j} \text{weight}(x)$    (weighted error on training set).
5.      If $\epsilon_i > 1/2$, set $S'$ to a bootstrap sample from $S$ with weight 1 for
            every instance and goto step 3 (this step is limited
            to 25 times after which we exit the loop).
6.      $\beta_i = \epsilon_i / (1 - \epsilon_i)$
7.      For-each $x_j \in S'$, if $C_i(x_j) = y_j$ then $\text{weight}(x_j) = \text{weight}(x_j) \cdot \beta_i$.
8.      Normalize the weights of instances so the total weight of $S'$ is $m$.
9.  }
10. $C^*(x) = \arg\max_{y \in Y} \sum_{i : C_i(x) = y} \log \frac{1}{\beta_i}$
```

Output: classifier C^*.

The AdaBoost algorithm combines several induction algorithms, so that in the presence of a new instance, it tries to determine the class/label of this instance as a function of the decisions (votes) made by all these algorithms.

This paper presents a method to extract the most relevant sentences from an original document using AdaBoost which is also called AdaBoost.M1. This algorithm produces a set of classifiers and assigns a weight to each one of them. AdaBoost also modifies the training instances weights delivered as input to each classifier on the basis of the models formerly constructed.

The aim of the process adopted by AdaBoost is to minimize the error over the different inputs distribution. The different parameters of the algorithm are:

T: number of iterations
S_1,S_2,...,S_t : t weighted training set
C_1,C_2,...,C_t: t classifiers
x_i is the input vector of the training instance number i
y_i is the label of the training instance number i

For more details about the algorithm we refer the reader to [3] paper.

3.2 System Design

Our architecture is based on two phases: the first one aims to build the AdaBoost learning model. In the second phase, this model which was produced in the previous phase is tested by producing a summary using the model and assessing its quality.

Stage One: Building the Learning Model

The general idea of this process is to construct the training data from a parallel corpus of pairs of <source, summary>documents. This process is presented in more detail in Algorithm 1.

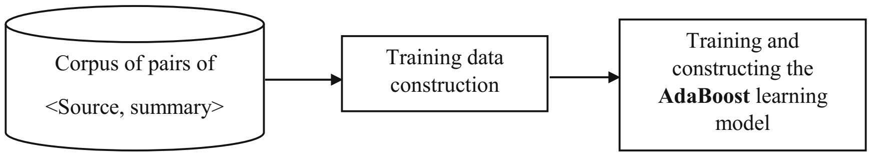
Corpus of pairs of
<Source, summary>
Training data construction
Training and constructing the **AdaBoost** learning model

Fig. 1. Stage one: Building the learning model

Algorithm 1.

```
1: Input C: corpus of pairs <source, summary >of documents
2: Output T: Training data
3: T=Null {The training set is initialy empty}
4: for each pair <$D_i$,$D_j$ >in C do
5:    $D_i$:=Segments ($D_i$)
6:    $D_j$:=Segments ($D_j$)
7:    for each sentence $S_i$ in $D_i$ do
8:       if $S_i$ exists in $D_j$ then
9:          $S_i$.label:=1
10:      else
11:         $S_i$.label:=0
12:      end if
13:      $F_i$:=Extract_features ($S_i$)
14:      $T_i$:=CreateInstance ($F_i$, $S_i$.Label)
15:      T.add($T_i$)
16:   end for
17: end for
18: Return T
```

Building the data that will be used to construct the classifier is the cornerstone of our proposal. Given a set of training documents which is a corpus of pairs <source, summary>. The process starts by segmenting each pair of documents <d_i,d_j>from the training corpus which gives us a document $d_i = \{s_{i1}, s_{i2},..,s_{im}\}$ and a document $d_j = \{s_{j1}, s_{j2},..,s_{jn}\}$ each of them containing a set of sentences. The second step is to find for each sentence in the source document whether it has been removed or kept in the corresponding summary. Depending on the result, we assign a value of one (1) as a label to the sentence to indicate that the sentence has been kept and zero (0) to indicate that the sentence has been deleted in the summary document. The function extract_features is responsible to get the different features from a given sentence. The different features considered in this work are: the number of words in a sentence that are common with those in the title; a discrete function which gives the value zero (0) in case the sentence is not the first in the text and one (1) otherwise; the position of the sentence in the document (the first sentences of the article are most of the time informative); The number of keywords in the sentence; The length of the sentence; a function that returns zero (0) if the sentence is the last one in the text and one (1) otherwise. Then, given the label as well as the different features, these parameters as passed as arguments to the function CreateInstance. Thus, a new instance is created and added to the training data. The process is repeated for each pair of documents until we get the entire training data.

Stage Two: Summary Construction

The process depicted in figure 2 is responsible for the generation of a summary from a source document. The different procedures used in this process are described in more detail in Algorithm 2.

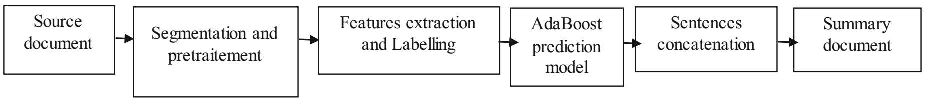
Source document
Segmentation and pretraitement
Features extraction and Labelling
AdaBoost prediction model
Sentences concatenation
Summary document

Fig. 2. Stage two: summary construction

Algorithm 2.

```
 1: Input $D_i$ (Source document)
 2: Output $D_j$ (Summary made by the system)
 3: $D_i$:=Segments ($D_i$)
 4: $D_j$:=Null {The summary is initialy empty}
 5: for each sentence $S_i$ in $D_i$ do
 6:     $F_i$:=Extract_features($S_i$)
 7:     Decision:=AdaBoost_predict($F_i$) { Get the decision about $S_i$ from the AdaBoost
        model}
 8:     if decision=1 then
 9:         $D_j$:=$D_j$+$S_i$ { Concatenation between sentences }
10:     end if
11: end for
12: Return $D_j$
```

The algorithm takes as input a source document (D_i) then applies a set of operations to generate a summary (D_j). The system starts with the initialization of the summary document (D_j) by associating an empty value to it. Then it segments the source document into sentences and performs the pretreatment specific to the Arabic text that we will explain latter. Once the documents get segmented, for each sentence (S_i) in the source document (D_i), the different features are extracted from the sentence (S_i). Next, the extracted features are passed to the AdaBoost classifier which decides whether the sentence should be included in the summary. If so, this sentence is concatenated to the summary under construction. This process is repeated until the complete summary is generated.

4 Implementation and Test

4.1 Corpus

Machine learning allows to learn rules from the observation of a set of instances and their corresponding labels. In this case, a corpus of source documents and their corresponding summaries was used. The instances are the feature vectors of the sentences in a document, and the labels are the decisions about the importance of the sentence: label zero(0) indicates that the sentence is not important and one (1) indicates that the sentence is important and should be included in the summary.

We have built a corpus of documents about technology news. These articles have been collected from the websites cnnarabic.com and bbcarabic.com . The total number of documents is 20. This dataset is a parallel corpus of <source, summary>pairs. The summaries were manually produced.

4.2 Pretreatment and Normalization of the Arabic Text

An initial step in our system is the normalization and pretreatment of the text. Most of the operations that we will present in the sequel are specific to the Arabic language. We have performed the followings:

– Words encoding: We have used UTF-8 to encode the different characters in the documents.
– Removal of vowels: the various diacritics have been removed.
  Example: The word الْعَرَبِيَّةُ becomes العربية
– Removal of Elongation: it is purely aesthetic and it has nothing to add to the meaning of the word.
  Example: The word العـــــربية becomes العربية
– Normalization of "HAMZA": the following letters: ALEF_HAMZA_ABOVE, ALEF_MADDA, HAMZA_ABOVE, BELOW ALEF_HAMZA, and BELOW_HAMZA are transformed to ALEF by removing the Hamza.
  Example: أهؤلاء becomes اهؤلاء

4.3 Comparison

As we have shown in section 2, Arabic text summarization is not studied enough in the literature. Moreover, we can state that it is very difficult to compare our technique against other existing techniques, due to the lack of gold standard corpora on the one hand and the different measures used to assess text summarization on the other. This is why we have decided to compare AdaBoost against the results obtained using multilayer perceptrons (MLP) and j48 decision trees. These techniques have been very successful for many AI applications. Table I below shows the results of the comparison between a multilayer perceptrons, j48 decision trees and the AdaBoost algorithm in terms of F1 score. In this setting, AdaBoost was used to boost the support vector machine classifier, which is a robust machine learning classifier introduced by Vapnik[15]. We have used 10-fold cross validation for all the machine learning algorithms.

Precision reflects the ratio of sentences extracted by the system and which were judged to be relevant.

Recall reflects the ratio of relevant sentences that the system extracted (i.e. did not miss).

We use TP to denote true positives and FP to denote false negatives. The recall(R), precision(P) and F1 score(F1) are calculated as follows.

$$P = \frac{TP}{TP + FP} \tag{1}$$

$$R = \frac{TP}{TP + FN} \tag{2}$$

F$_\beta$ score makes a balance between recall and precision; its general formula is given as follows:

$$F_\beta = (1 + \beta^2) * \frac{P * R}{\beta^2 * P + R} \tag{3}$$

F1 score: By setting the value of β to one we get

$$F_1 = 2 * \frac{P * R}{P + R} \tag{4}$$

Table 1. Comparaison between AdaBoost, J48 and MLP in terms of F1 score

Approach	**MLP**	**AdaBoost**	**j48**
F1-score	66,4%	66,60%	63.20%

4.4 Discussion

From the above results we conclude that AdaBoost has given better results than MLP and j48 decision trees in terms of F1 score. This is due to the voting mechanism adopted by AdaBoost. If we look at the task of text summarization itself and its subjective definition, we conclude that 66.60% F1 score is considered as an acceptable result. We can also state that it is very difficult to compare the proposed approach (AdaBoost) to other existing systems for various raisons. Among these, we can mention the following:

1. It is difficult to compare the performances of the approaches proposed for Arabic text summarization, because each work uses a different dataset.
2. Various researchers use different evaluation measures, which is due to the difficulty of the task itself. In fact, there is a lot of research underway developing new metrics to assess the quality of a summarization system [10,8]. In fact, the evaluation of text summarization systems is still an open issue that has to be tackled.
3. The community working on Arabic natural language processing, and specifically Arabic text summarization, is still quite small.
4. The complexity of the Arabic language in terms of its spelling, vocabulary as well as its morphology makes lexical, syntactic, and semantic ambiguity even higher.
5. The problem of diacritics (tashkyl) in the Arabic language makes it even more complex to handle.

5 Conclusion

We have presented in this paper a machine learning-based approach to Arabic text summarization that uses the AdaBoost algorithm. We have shown the results of the evaluation of the implementation using the F1-score measure. We have shown that this approach outperforms other existing machine learning systems in terms of the F1-score.

We envision a further development of this work in several directions. One direction would be to test our approach on an extended corpus that contains more documents. We also intend to introduce more features like part-of-speech tagging and semantic relations between sentences. Another interesting area of research is be to propose a new approach that treats the problem of text summarization but using an abstractive paradigm.

References

1. Abdel Fattah, M., Ren, F.: Probabilistic neural network based text summarization. In: International Conference of Natural Language Processing and Knowledge Engineering, NLP-KE 2008, pp. 1–6. IEEE (2008)
2. Azmi, A., Al-thanyyan, S.: Ikhtasir—a user selected compression ratio arabic text summarization system. In: Proceeding of International Conference of Natural Language Processing and Knowledge Engineering (NLP-KE 2009), pp. 1–7 (2009)

3. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36(1-2), 105–139 (1999)
4. Douzidia, F.S., Lapalme, G.: Lakhas, an arabic summarization system. In: Proceedings of DUC 2004(2004)
5. El-Haj, M., Kruschwitz, U., Fox, C.: Multi-document arabic text summarisation. In: 2011 3rd Computer Science and Electronic Engineering Conference (CEEC), pp. 40–44. IEEE (2011)
6. Freund, Y., Schapire, R.E., et al.: Experiments with a new boosting algorithm. In: ICML, vol. 96, pp. 148–156 (1996)
7. Keskes, I., Boudabous, M.M., Maaloul, M.H., Belguith, L.H.: Étude comparative entre trois approches de résumé automatique de documents arabes. In: Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, Grenoble, France, vol. 2, pp. 225–238. ATALA/AFCP (2012)
8. Lin, C.-Y.: Rouge: A package for automatic evaluation of summaries. In: Text Summarization Branches Out: Proceedings of the ACL-2004 Workshop, pp. 74–81 (2004)
9. Lloret, E., Palomar, M.: Text summarisation in progress: a literature review. Artificial Intelligence Review 37(1), 1–41 (2012)
10. Louis, A., Nenkova, A.: Automatically assessing machine summary content without a gold standard. Computational Linguistics 39(2), 267–300 (2013)
11. Nenkova, A., McKeown, K.: Automatic summarization. Foundations and Trends in Information Retrieval 5(2-3), 103–233 (2011)
12. Saggion, H., Poibeau, T.: Automatic text summarization: Past, present and future. In: Multi-source, Multilingual Information Extraction and Summarization, pp. 3–21. Springer (2013)
13. Schapire, R.E.: The strength of weak learnability. Machine Learning 5(2), 197–227 (1990)
14. Jones, K.S.: Automatic summarising: The state of the art. Information Processing & Management 43(6), 1449–1481 (2007)
15. Vapnik, V.N.: The nature of statistical learning theory. statistics for engineering and information science. Springer-Verlag, New York (2000)