Improved approaches for extracting Arabic Keyphrases

Mahmoud Nabil, Mohamed Aly, Amir Atiya

Computer Engineering, Cairo University, Giza, Egypt

Abstract. Keyphrases extraction has a considerable importance in many applications such as search engine optimization, clustering, summarization, and sentiment analysis. The importance of keyphrases comes from the semantic meaning they provide as they can be used as descriptors for the documents. In this paper we propose an approach for extracting keyphrases from Arabic documents using Arabic natural language processing tools (e.g. stemmer and part of speech tagger) to generate candidates that are weighted to select the predicted keyphrases.We compare our approach to the KP-Miner keyphrase extraction system, and show that it achieves comparable performance while being more flexible and more widely applicable.

1 Introduction

Keyphrases are the set of data that describes and gives information about documents. They are considered as descriptors that summarize and provide a brief summary for a given document. The number of documents on the Internet grow enormously every day (e.g. Wikipedia and Facebook), so the need for an automated system to extract these keyphrases increases. Some of the uses of the keyphrases such as: (1) Document Indexing : where the goal is to find the data items that enhance information retrieval systems. (2) Document Summarization: where the goal is to provide a brief description for the document. (3) Sentiment analysis: where the goal is to mention the main aspect intended by the sentiment. (4) Documents clustering: where the goal is to group documents by keyphrases or keywords. Despite the importance of the keyphrases and keywords most of the on-line documents don't have keyphrases attached to them. Moreover, the process of manually annotating keyphrases to a given document is a tough and time-consuming task. Therefore finding a way to automate the keyphrases extraction process is beneficial for many applications. In this paper we propose a preliminarily method that automates this process for Arabic documents by using some Arabic natural language processing tools that were built for that purpose. The system uses a prebuilt stemmer and Part of Speech (POS) tagger for the Arabic text as a preprocessing step to generate candidates for the token frequency inverse document frequency (TF-IDF) weighting algorithm. Our method can be adapted easily to work in different domains and extednded for languages other than Arabic by using the convenient preprocessing tools.

The contributions of this work can be summarized as follows:

1. A Hyprid technqiues for Arabic keyphrases extraction that merge linguistic and statistical features of the input document.
2. A simple Arabic light stemmer and part of speech tagger that were trainned on (Penn. Arabic Tree Bank).

2 Related Work

Two dominant techniques were considered for keyphrases extraction, which are unsupervised learning techniques such as n-grams weighting methods and supervised machine learning techniques. Supervised techniques have two main different approaches: keyphrase assignment and keyphrase extraction. In keyphrase assignment [2] a predefined list of keyphrases is used then a classifier for each keyphrase is built such that a given document is calssifed positively if it contains this keyphrase. In contarst the keyphrase extraction approach does not have a predefined list of keyphrases but instead utilizes lexical, statistical and linguistic information to identify the keyphrases in the document. In [9,10] the author tested two approaches for this task: the first approach used a general-purpose C4.5 algorithm while the second approach introduced GenEx (Genitor and extractor) algorithm. In C4.5 two classes were used (keyphrase and non-keyphrase) then the author studied the effects of changing the number of trees, changing the ratio of the classes, and changing the size of each random sample. In GenEx Turney used a genetic algorithm (Genitor) to adapt a set of heuristic rules used by the (Extractor). However, when the best set of heuristic rules are known the Genitor can be discarded. Turney shows that using specialized knowledge for keyphrases extraction performes significantly better than the general-purpose C4.5 algorithm. In [11] the authors adapted Turney supervised findings by introducing *KEA* (Keyphrase extraction algorithm). KEA system employs a supervised Naïve Bayes model to extract unseen keyphrases from a given document. KEA uses two main features: the TF-IDF, and the relative distance within the document. In [7] the author used semantic networks to model the training documents where the structure and the dynamics of these networks were used to get the keyphrases. In [5] the author used graphs to represent sementic relationships among phrases in the document then a community detection algorithm that identify the group of vertices that are related to the main topic of the document. The algorithm has the advantage of clustering of the keyphrases beside identifying them.

Regarding Arabic little work has been proposed to target automatic keyphrases extraction due to the lack of avialable Arabic datasets for this task. In [3] the authors developed an algorithm that uses a set of heuristc rules such as the number of times and the position where the keyphrase first appears in the document, then they used a modefied TF-IDF weight calculation formula induced from the statistics of the document itself. In [4] the authors proposed a pure supervised learning technique that uses some statistical and linguistic aspects as a feature vector to the system learning model, also they used a sample of 30 documents that are manually reviewed and annotated to train their model. Another Arabic system is Sakhr Keyword Extractor but the system is commercial and no technical details about the system are published.

3 System describtion

In this work, we believe that combining linguistic methods with statistical methods will lead to better performance. Towards this end we decided to combine both methods by using a stemmer and Part of Speech tagger (POS) that were built for this purpose. The stemmer and the POS tagger help in generating a list of candidate patterns. These patterns are then weighted using TF-IDF algorithm to extract a list of keyphrases for the input document. Our work is inspired mainly from the following observations:

– Keyphrases appear in some linguistic patterns such as nouns, proper nouns, noun then adjectives and similar patterns. In this work, we utilize this fact by using a POS tagger that can capture these patterns.
– Classical techniques for information retrieval such as TF-IDF still have high performance compared to most of the current used methods.
– Although the TF-IDF algorithm gives low weight to the unimportant keyphrases, the use of stop word lists is very beneficial for Arabic language as some stop-words in Arabic are compound ones and do not occur frequently.
– Linguistic features boost the performance of most natural language processing applications as they add abstract semantic representation to the input text.

The proposed system employs the previous observations using two main execution steps: candidate keyphrase selection, TF-IDF weight calculation. The following sections explain the two steps in more detail.

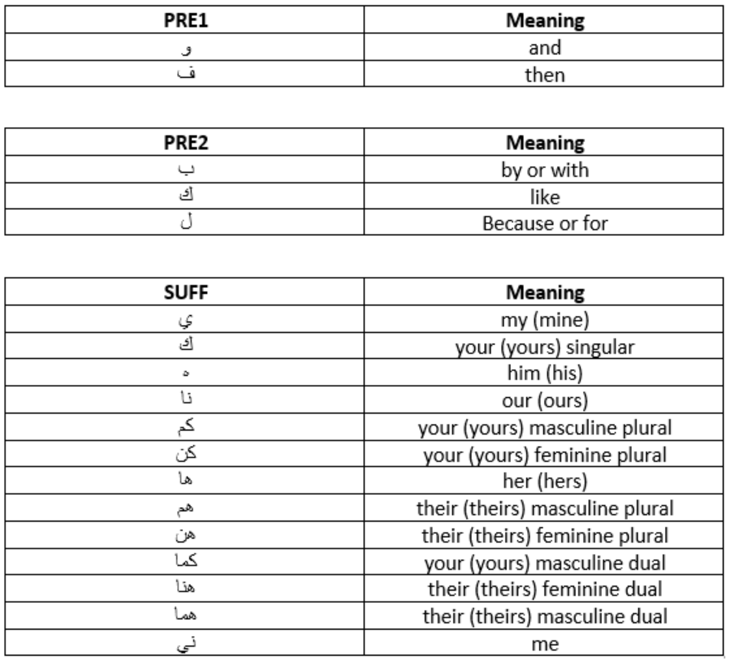
4 Stemmer and POS tagger

Keyphrases extraction mainly depends on a very few set of POS tags that can affect its performance such as nouns, proper nouns and other set of tags. This set of tags is crucial in keyphrase extraction so we decided to implement our own stemmer and POS tagger mainly to focus on just a small set of tags that are needed for this task. The other reason was to make compatible modules that can be managed and maintained easily. Our results are not directly comparable to [1] and [6] as they used fewer training parts from Penn. Arabic Treebank.

We developed a stemmer and a POS tagger. Both modules are trained on three parts from (Penn. Arabic Treebank): part1 (ATB1), part2 (ATB2), and part3 (ATB3). Penn. Arabic Treebank [8] is the largest Arabic tree bank developed by the Linguistic Data Consortium (LDC). It consists of about one million words in different articles covering various topics such as sports, politics, news, etc.

In order to test the tokenizer and the POS tagger thoroughly, we partition the data into training and test sets, the ratio among these two sets is 8:2 respectively.

4.1 Tokenizer



PRE1	Meaning
و	and
ف	then

PRE2	Meaning
ب	by or with
ك	like
ل	Because or for

SUFF	Meaning
ي	my (mine)
ك	your (yours) singular
ه	him (his)
نا	our (ours)
كم	your (yours) masculine plural
كن	your (yours) feminine plural
ها	her (hers)
هم	their (theirs) masculine plural
هن	their (theirs) feminine plural
كما	your (yours) masculine dual
هنا	their (theirs) feminine dual
هما	their (theirs) masculine dual
ني	me

Fig. 1. The set of prefixes and suffixes and their meanings

The tokenizer module uses an SVM classifier that takes an Arabic text without any processing as input, and assigns a tag to each letter according to (IOB) labeling scheme that is used in many NLP tasks. The IOB labeling consists of three essential tags: I (inside), O (outside), and B (begin). The tagset used by the tokenizer is {B-PRE1, B-PRE2, B-WRD, I-WRD, B-SUFF, I-SUFF}. Where PRE denotes prefix, WRD denotes word, and SUFF denotes suffix. Figure 1 shows the complete set of prefixes and suffixes used by the tokenizer. Note that the Arabic word can have up to two levels of prefixes that precede it and at most one suffix that follows it. The two level of prefixes are mutually exclusive sets where the first level of prefixes is assigned the tag B-PRE1 while the second level is assigned the tag B-PRE2.

4.2 Stemmer Evaluation

Table 1 shows the experimental results of the stemmer. The results are almost perfect due to the fact that the dataset is huge and a large percent of words are not tokenized at all.

Classifier	Precision	Recall	Accuracy
SVM	0.99735	0.99815	99.76%

Table 1. Tokenization experimental results

4.3 POS Tagger

POS TAG	Meaning
CD	Number
NN	Noun
NNP	Proper Noun
NNS	Noun (plural)
NNPS	Proper Noun (plural)
JJ	Adjective
RB	Adverb
CC	Conjunction
DT	Determinant
RP	Particle (Except- Negation ...)
VBP	Verb Imperfect
VBN	Verb Passive
VBD	Verb Perfect
VB	Verb Imperative
UH	interjection
PRP	Pronoun
PRP$	Possessive Pronoun
WP	Relation Pronoun

Table 2. The POS tagger tagset

POS tagging is the task of annotating each word in the input text with a tag that identify its definition and the context of that definition. Since Arabic is a morphological rich language many words can take different POS tags in different contexts. For example the Arabic word (ktb: transliterated according to Buckwalter transliteration scheme[1]) may have the English meaning (write) in one context so it will be considered as verb or the English meaning (books) in another context so it will be considered as noun. Our POS tagger is based on [1] experiments where they used the Reduced Tag Set (RTS) of the Penn. Arabic Treebank. Table 2 shows the tag set (RTS) used by our POS tagger and the meaning of each tag. The reason for choosing the (RTS) tagset is to have a high precision in this module so it can be used safely in further sentiment analysis experiments. The features used by the SVM classifier are:

1. Window of context words of size -2 to +2 from the current token.
2. First N characters from the current token as N-grams where N<=4.
3. Last N characters from the current token as N-grams where N<=4.

[1] http://en.wikipedia.org/wiki/Buckwalter_transliteration

Algorithm 1 TF-IDF Algorithm

```
TF-IDF(documents)
DF_Map={}
Token_Frequency_Map={{}}
for each document in documents :
    Tokens=Apply_Tokenization(review)
    Tags=Apply_Tokenization(tokens)
    Patterns=Get_Valid_Patterns(Tokens,Tags,self.StopWord)
    for each Pattern in Patterns
        Token_Frequency_Map[review][Pattern]+=1
    Patterns_Set=Set(Patterns)
    for each Pattern in Patterns_Set
        DF_Map[Noun]+=1
return DF_Map,Token_Frequency_Map;
```

4. Token contain alphabetical or numeric characters.
5. POS tag of the previous two tokens.

4.4 POS tagger Evaluation

Classifier	Precision	Recall	Accuracy
SVM	0.97	0.97	97.05%

Table 3. POS tagger experimental results

Valid single patterns	Valid compund patterns
NN	NN NN,NN NNP, NN NNS,NN NNP,NN JJ
NNS	NNS NNS,NNS NNP, NNS NNS,NNS NNP,NNS JJ
NNP	NNP NNS,NNP NNP, NNP NNS,NNP NNP,NNP JJ
JJ	-

Table 4. Valid POS tags patterns

Table 3 shows the experimental results of the POS. The results are high as expected because the (RTS) tagset is simpler than the extended tagset of the (Penn. Arabic Treebank) which contain about 131 tag.

4.5 Proposed algorithms

In this work we tried three different methods in the first method TF-IDF algorithm is used where the algorithm takes a set of documents as input and returns a score for each valid pattern in each document. We define a valid pattern as one of the patterns appearing in Table 4. The TF-IDF algorithm first extracts patterns from the documents then uses the definitions of term frequency and inverse document frequency, to produce a Tf-IDF weight for each valid pattern in the document. The most important feature of TF-IDF algorithm is that it down-weights the very frequent, and boosts the score of the more "informative" words. Also we used a stop word list of about 700 stopword gatherd from the internet[2] . The purpose of this list is to eleminate the Arabic low frequent stopwords that were missed by the TF-IDF algorithm. Algorithm 1 shows the pseudocode of the TF-IDF algorithm. In the second method we used **Google's Word2Vec**[3] library that used unsupervised deep-learning approach to produce semantic vectors for each corpus word as output. We trained **word2vec** using Wikipedia Arabic dump[4] to get the vector representation of the words then in the evaluation process we used the cosine similarity to measure the distance between the title of each document and the valid patterns in it. The third method is a combination between the first two methods where the combination formula is shown in equation 4.

$$TF-IDF_{Token,Document} = TF_{Token,Document} * IDF_{Token} \tag{1}$$

$$TF_{Token,Document} = Frequency(Token, Review) \tag{2}$$

$$IDF_{Token} = \frac{N}{DocumentFrequency} \tag{3}$$

$$Weight_{Token,Document} = TF-IDF_{Token,Document} * \frac{1}{CosineSimilarity(DocumentTitle, CandidatePattern)} \tag{4}$$

4.6 Proposed Algorithms Evaluation

As mentioned before, there are very few Arabic datasets for keyphrases extraction task so in order to evaluate our approach we used the same corpus used by the KP-Miner system [3]. The corpus consists of one hundred documents selected randomly from Wikipedia. Each document is associated with a *key file* that contains the keyphrases associated with that document. On average each document contains about 8 keyphrases. These keywords were acquired from the meta-tag of the original Wikipedia article. The average number of words for each document in this corpus is 804. We compare the proposed method with the KP-Miner on this corpus.

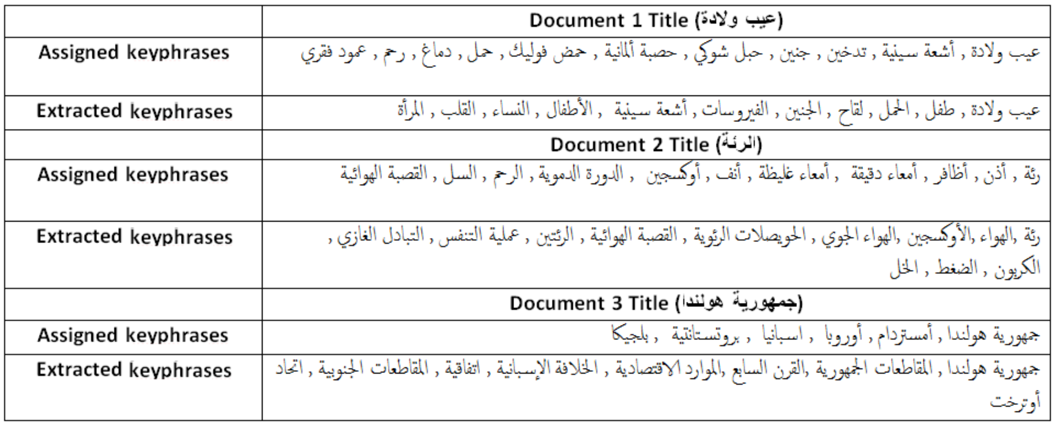
[2] http://goo.gl/MJA5V0
[3] https://code.google.com/p/word2vec/
[4] https://github.com/anastaw/Arabic-Wikipedia-Corpus

The results are shown in Table 5. We compared our results with the KP-Miner system using the same performance metrics of the KP-Miner which are the average precision, average recall, and the average number of the detected heyphrases by using the top 20 generated keyphrases.

	KP-Miner	**TF-IDF Algorithm**	**Word2Vec Algorithm**	**Hybrid Algorithm**
Avg. Precision ± SD	0.132 ± 0.062	0.112 ± 0.058	0.085 ± 0.046	0.102 ± 0.047
Avg. Recall ± SD	0.383 ± 0.248	0.349 ± 0.242	0.288 ± 0.249	0.314 ± 0.254
Avg. Detected Keys ± SD	2.490 ± 1.210	2.250 ± 1.160	1.700 ± 0.932	2.000 ± 0.957

Table 5. Comparison between the proposed methods and the KP-Miner

Despite our primlinary result being slightly lower than that of the KP-Miner system, our proposed system is still comparable to the other keyphrase exractor systems for the following reasons:



	Document 1 Title (عيب ولادة)
Assigned keyphrases	عيب ولادة , أشعة سينية , تدخين , جنين , حبل شوكي , حصبة ألمانية , حمض فوليك , حمل , دماغ , رحم , عمود فقري
Extracted keyphrases	عيب ولادة , طفل , الحمل , لقاح , الجنين , الفيروسات , أشعة سينية , الأطفال , النساء , القلب , المرأة
	Document 2 Title (الرئة)
Assigned keyphrases	رئة , أذن , أظافر , أمعاء دقيقة , أمعاء غليظة , أنف , أوكسجين , الدورة الدموية , الرحم , السل , القصبة الهوائية
Extracted keyphrases	رئة ,الهواء ,الأوكسجين ,الهواء الجوي , الحويصلات الرئوية , القصبة الهوائية , الرئتين , عملية التنفس , التبادل الغازي , الكربون , الضغط , الخل
	Document 3 Title (جمهورية هولندا)
Assigned keyphrases	جمهورية هولندا , أمستردام , أوروبا , اسبانيا , بروتستانتية , بلجيكا
Extracted keyphrases	جمهورية هولندا , المقاطعات الجمهورية ,القرن السابع ,الموارد الاقتصادية , الخلافة الإسبانية , اتفاقية , المقاطعات الجنوبية , اتحاد أوترخت

Fig. 2. Proposed system sample results

1. The proposed system employs a new keyphrase extraction technique where it combines the statistical and the linguistic information of the document.
2. The proposed system provides both the keyphrases of the given document together with their part of speech tags.
3. Some of the false positive keyphrases extracted by our system are acceptable keyphrase (See Figure 2) that human annotators may choose as important keyphrases. This is an indicator of the difficulty of this task, and that our performance will increase with better annotated datasets.

5 Conclusion and Future Work

In this paper we present a preliminarily method for extracting keyphrases from Arabic documents. The proposed method combines both linguistic and statistical information. We also introduce a stemmer and a POS tagger that are built for this purpose and trained on (Penn. Arabic Tree Bank). The system used the stemmer and the POS tagger to generate a list of candidate keypharases patterns that are weighted using TF-IDF algorithm. To evaluate the effectiveness of the used method the system was experimented on a corpus of Wikipedia articles. For further work to improve our results we plan to:

1. Try the system on more datasets.
2. Modify the weighting factor of the TF-IDF algorithm to take into account the position of the phrase in the document
3. Explore using Arabic base phrase chunking to enhance the patterns generating algorithm.
4. Experiment with the system on other languages.

References

1. Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. Automatic tagging of arabic text: From raw text to base phrase chunks. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 149–152. Association for Computational Linguistics, 2004.
2. Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155. ACM, 1998.
3. Samhaa R El-Beltagy and Ahmed Rafea. Kp-miner: A keyphrase extraction system for english and arabic documents. *Information Systems*, 34(1):132–144, 2009.
4. Tarek El-Shishtawy and Abdulwahab Al-Sammak. Arabic keyphrase extraction using linguistic knowledge and machine learning techniques. *arXiv preprint arXiv:1203.4605*, 2012.
5. Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th international conference on World wide web*, pages 661–670. ACM, 2009.
6. Nizar Habash and Owen Rambow. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 573–580. Association for Computational Linguistics, 2005.
7. Chong Huang, Yonghong Tian, Zhi Zhou, Charles X Ling, and Tiejun Huang. Keyphrase extraction using semantic networks structure analysis. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 275–284. IEEE, 2006.
8. Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. The penn arabic treebank: Building a large-scale annotated arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, pages 102–109, 2004.
9. Peter Turney. Learning to extract keyphrases from text. 1999.
10. Peter D Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.

11. Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255. ACM, 1999.