Leveraging Arabic Morphology and Syntax for Achieving Better Keyphrase Extraction

Muhammad Helmy, Dario De Nart, Dante Degl'Innocenti and Carlo Tasso
Artificial Intelligence Lab
Department of Mathematics and Computer Science
University of Udine, Udine, Italy
Email: {alameldien.muhammad, deglinnocenti.dante}@spes.uniud.it, {dario.denart, carlo.tasso}@uniud.it

***Abstract*—Arabic is one of the fastest growing languages on the Web, with an increasing amount of user generated content being published by both native and non-native speakers all over the world. Despite the great linguistic differences between Arabic and western languages such as English, most Arabic keyphrase extraction systems rely on approaches designed for western languages, thus ignoring its rich morphology and syntax. In this paper we present a new approach leveraging the Arabic morphology and syntax to generate a restricted set of meaningful candidates among which keyphrases are selected. Though employing a small set of well-known features to select the final keyphrases, our system consistently outperforms the well-known and established systems.**

***Keywords*-Arabic Natural Language Processing; Keyphrase Extraction; Lemmatization; Stemming;**

I. Introduction

Arabic digital content in the recent years has grown considerably on the Web pushed by the increasing access of Arabic countries to the internet and social media. To tackle the approaching content flood Natural Language Processing (NLP) tools in the likes of the ones already available for western languages, such as English, are extremely desirable. The Arabic language, however, presents some major differences from western languages that pose severe limits to the performance of algorithms and techniques originally designed for these languages.

A Keyphrase (KP) is commonly defined as a short phrase typically consisting in one to three words representing an entity or a concept that is somehow representative of the content of a given text. The field of automatic Keyphrase Extraction (KPE) has been widely explored in the literature. The most widespread approach to KPE consists in generating a pool of candidate KPs (CKPs) and then select the most relevant ones according to a set of features. Each of these steps can be performed in several different ways, however most approaches are designed for English and then tuned to work on other languages.

In this paper we present a novel technique for Arabic KPE specifically designed for the Arabic language. The claim of this work is twofold: we believe that approaches tailored on the key characteristics of non-western languages could provide better results than just tuning existing systems originally designed for western languages; secondly, we also believe that a more accurate CKP generation phase, avoiding generation of clearly non-relevant phrases, coupled with a relatively simple selection phase could provide better results than a complex candidate selection relying on a wide array of features coupled with a naïve CKP generation phase.

II. Related Work

Many authors have investigated Arabic KPE in the literature, however, most solutions consist in fitting techniques developed for western languages. The most well-known system in this field is KP-Miner [1], which is built on purpose to provide satisfactory results in both Arabic and English in an unsupervised fashion. KP-Miner, albeit relying on purely statistical principles, is effective and is regarded in the literature as the gold standard to compare to.

The authors of [2] and [3] adopt two approaches that select CKPs according to a set of linguistic features derived from Part-Of-Speech (POS) analysis. The inclusion of such linguistic features provides a significant accuracy increase as observed in the English language in [4].

The approach of term equivalence classes presented in [5] includes a *filtering* phase which removes some CKPs according to linguistic knowledge, and then groups terms into equivalence classes according to their roots. The evaluation is then performed on an ad-hoc built human annotated test corpus upon which the authors claim to achieve significantly higher precision and recall than KP-Miner.

The authors of [6] and [7] pinpoint the most notable features of Modern Standard Arabic (MSA) from a NLP point of view. First of all, Arabic is strongly agglutinative language, like German, allowing for the aggregation of whole phrases in one word, and also allows the dropping of subject pronouns which means that not all sentences may have a subject, like Italian, Spanish, and Korean. Moreover, in Arabic there is no letter capitalization, letters may change according to their position, and the usage of punctuation is minimal, making sentences hard to recognize. Furthermore, Arabic is highly ambiguous due to the presence of homographs and the complex internal structure

978-1-5090-0922-0/16/$31.00

Table I: Example of Arabic word contains five tokens.

Word	أنلزمكوها				
Translation	Shall we compel you to accept it				
Tokens	ها	كم	لزم	ن	أ
Translation	it	you	Compel to accept	We	Shall

Table II: Examples of words with different lemmas

Word	Translation	Lemma	Translation
الكتب	The books	الكتب	Book
كتبة	Writers	كاتب	Writer
المكتبات	The libraries	مكتبة	Library
مكاتب	Offices	مكتب	Office
مكاتبات	Correspondences	مكاتبة	Correspondence

of sentences that makes their interpretation highly context-dependent. Finally, Arabic is a relatively free word order language, implying that phrase patterns may frequently vary. All these characteristics make Arabic unique and suggest that techniques that do not take them into account may achieve poor performance to the eyes of a proficient Arabic speaker.

III. The Proposed Approach

Our method relies on a simple principle: avoiding the generation of CKPs that do not make sense. Aside from that it consists in a traditional KPE process that computes for each CKP a score based upon simple distributional information. Spotting meaningless candidates is trivial for a human, but it is not for a machine, unless some basic acceptability rules are coded into the candidate generation algorithm. The phases of the proposed approach are described in the following subsections.

A. Text Splitting and Segmentation

Dividing Arabic text into sentences and tokens is not an easy task. Punctuation marks and whitespaces do not define the boundaries of sentences and words precisely like English. Moreover, using punctuation marks in MSA is optional and they are rarely used in a strict manner. Additionally, a single word can hold a complete sentence or a set of concatenated tokens (see table I). The first step of this phase is to segment and tokenize the text into its atomic tokens. This task is performed using Stanford Arabic Segmenter [8]. Following that, the segmented text was split into sentences which will be parsed to find out the Noun Phrases (NPs). The splitting process used the punctuation marks of period, question mark, exclamation mark and colon as end marks for sentences. But since these punctuation marks are not strictly used in Arabic scripts, the outcome contained lengthy sentences. To overcome this problem, we leveraged the fact that most of the phrases in MSA are of the structure Verb-Subject-Object (VSO) [6]. So, the outcome sentences were POS-tagged (see Section III-B) and then split into smaller sentences using verbs as markers for sentence boundaries.

B. Text POS-tagging and Parsing

Every token in the segmented text was assigned a POS-tag depending on its location and context. The token may be identified as noun, verb, adjective, preposition, etc. The text is POS-tagged using Stanford Arabic Part of Speech Tagger [9] and employing the Penn Arabic Treebank tag set [10]. POS-tagged text is used to determine CKPs and sentences boundaries. After that, the text is parsed to detect and generate a list of all NPs. Every NP contains one or more word and it is found between the parenthesis of NP in the parsed text. Some NPs may be sub-phrase of another NP.

C. Lemmatization and Tokens Grouping

The related tokens of the text should be grouped using their equivalent linguistic form (LF). All of the Arabic KPE systems are based on stemming like western languages.

According to [11], [6], Arabic has a rich morphological structure, so it requires a different stemming process from other languages. In Arabic, stemming (whether it's root-based or light) reduces the words to their corresponding root or stem which is almost three-character word. This sharp process can cause errors in automatic translation, text clustering, text summarization [11], [12], and also for KPE. On the other hand, Lemmatization is a normalization approach that finds the nearest dictionary form for a given word (called lemma). Since Arabic has a rich derivative rules for forming the words, stemming always gives one root for a huge number of words which in sequence produces over-semantic grouping for unrelated words. On the contrary, lemmatization groups only words which have similar dictionary look-up entry without over-reducing the words. For example, in table II, all of the five words have the same root (كتب, Katab, Write), but have different lemmas. The segmented text was lemmatized using Aramorph [13].

After lemmatizing the text, a list of Linguistic Lexemes (LLs) is generated. Every entry in LLs consists of a set of atomic tokens with the same lemma. Table III introduces an example of an LL containing five tokens with different POS-tags but with the same lemma (حرية, Hurria, Freedom). The last column of the table represents the number of times the token has occurred in the text. The representative text of an LL is the token with the highest number of occurrences, therefore the representative text for the LL in the table is the token " حرية" in the second row.

D. Possible CKPs Identification

Initially, all the LLs Which are nouns or adjectives and not stopwords are added to the CKPs list as uni-gram KPs. Then, the

Table III: An example of linguistic lexeme structure

Token	Translation	Lemma	POS	No. Of Occurrences
الحرية	The freedom	حرية	DTNN	3
حرية	Freedom	حرية	NN	10
الحريات	The freedoms	حرية	DTNNS	7
حريات	Freedoms	حرية	NNS	2

regular expression: *(NOUN|ADJECTIVE) ((CONNECTOR)?(NOUN|ADJECTIVE)){1,n-1}* is used to search the POS-tagged text to find the n-gram CKPs with n greater than one. The connector can be a coordinating, preposition, or subordinating conjunction. The maximum length of KP (n) is supplied to the system as a number greater than zero or can be left as zero, in this case the system will not put an upper limit to the length of CKPs. When the system detects a CKP, it checks whether the text of CKP forms a single NP in the NPs list or not. If the check succeeds, the CKP will be added to CKPs list along with references to its components from LLs. During our tests we observed that in average the NP check decreases the number of CKPs by roughly 50%.

E. Ranking CKPs

After NP check, the generated CKPs are scored and ranked to return the top-k KPs where k is passed to the system as an integer parameter.

CKPs are scored according to the following simple score equation (SC):

$$SC(CKP) = \begin{cases} \frac{\#LF\_Occ(CKP)}{\#DocTerms} & Length(CKP) = 1 \\ \frac{\#LF\_Occ(CKP) + \sum_{k=1}^{n} SC(LGram_k)}{Length(CKP)} & Length(CKP) > 1 \end{cases}$$

Where $LGram_k$ is the kth legal sub-KP of the considered CKP. Some long CKPs may contain more than one LGram or sub-KP. For example, a CKP of (ثورات الربيع العربى, The Arab Spring Revolutions) has three LGrams; {(ثورات, Revolutions), (الربيع, The Spring), (الربيع العربى, The Arab Spring)}. $Length(CKP)$ is the number of all possible LGrams of the CKP. $\#LF\_Occ(CKP)$ is the number of occurrences of the LFs of the considered CKP. If $Length(CKP) = 1$, then CKP contains only one single-token LGram and $\#LF\_Occ$ of that CKP will equal to summation of the number of occurrences of its related LL. For instance, $\#LF\_Occ$ of the CKP of (الحرية, The Freedom) is 22 which is the summation of its LL occurrences (Table III). $\#DocTerms$ is the total number of terms in the document, The recursive step is needed since there is no upper limit in Arabic to the nesting of phrases and a legal sub-KP could include its own legal sub-KPs.

IV. Experimental Evaluation

A testbed system was developed and benchmarked against other Arabic KPE approaches on three datasets already used in related works.

Table IV: Evaluation datasets details

Dataset	Topic	# of docs	Avg. Size in words	Avg. # of KPs
DS1	Leadership and management	27	1227	7.8
DS2	General Wikipedia pages	100	776	7.9
DS3	Agriculture, environment, and food	35	641	11.1

Table V: Comparison between the proposed system and other approaches.

	KP-Miner	TF-IDF	Word2Vec	Hybrid	Our System
Avg. Precision	0.13 ± 0.06	0.112 ± 0.06	0.09 ± 0.05	0.10 ± 0.05	0.13 ± 0.08
Avg. Recall	0.33 ± 0.25	0.349 ± 0.24	0.29 ± 0.25	0.31 ± 0.25	0.37 ± 0.25
Avg. Detected Keys	2.49 ± 1.21	2.250 ± 1.16	1.70 ± 0.83	2.00 ± 0.93	2.53 ± 1.52

A. Evaluation Datasets

All of the existing Arabic KPE approaches have been tested and evaluated against datasets built by their authors. These datasets were built by collecting web published documents from news portals, wikisites, and scientific articles. After that, KPs are assigned to the document collection manually by the authors or some experts. There is no a gold standard dataset for testing and training Arabic KPE systems. We decided to not build a custom dataset to avoid bias. Instead, we used three datasets already known in the literature and described in [1], [14] and [3]. Table IV presents the key characteristics of the considered datasets.

B. Experimental Results

Different experiments have been conducted to assess the performance of the proposed approach with the state of the art. In the first experiment, we benchmarked our system against four approaches. The considered baselines are: KP-Miner, a hybrid method based upon KP-Miner, a distributional approach based on Google's Word2Vec library, and *Term Frequency–Inverse Document Frequency* (TF-IDF) [1], [14]. For each system and dataset, we evaluate precision, recall and the average number of correctly extracted KPs per document; and for all these measures, the mean value and the standard deviation (± SD) are provided. This benchmark was performed over DS2 (see table V).

We can observe how the proposed approach outperforms the baseline ones in terms of correctly extracted KPs per document with an average of 2.53. This result is remarkable considering the characteristics of the DS2 dataset where the number of human annotated KPs per document can vary between 1 and 12.

The second experiment was performed to compare the results of extracting KPs using the lemmatization

Table VI: Comparison between Arabic-KEA using stemmers and our approach with lemmatizer

Dataset	Statistical stemmer	Rule based stemmer	Lemmatizer
DS1	1.56±1.59	0.67±10	2.78 ±1.3
DS2	2.58±1.24	1.17±0.94	3.75 ±1.42
DS3	1.4±0.86	0.96±0.87	2.57 ±1.67

approach which is employed by our system and the stemming approach which is adopted by Arabic-KEA system [3]. Arabic-KEA is a framework for KPE from Arabic news documents and it is based on the KEA [15]. Since KEA is an open software, it has encouraged many researchers to adapt it to other languages .

Arabic-KEA uses two different approaches for stemming: statistical and rule-based stemming. The two systems were run on the three datasets and the average number of detected KPs was computed. The results shown in Table VI suggest that lemmatization consistently produces more correct KPs than stemming.

V. Conclusion and Future Work

This article has presented an approach for Arabic KPE utilizing the specific features of morphology and syntax of the language. The morphology feature emerged in using lemmatization which helps in fair grouping for related words with the same lemma. Using syntax rules, a set of contiguous words are considered as a legal CKP if and only if they form a complete NP. NP-feature decreases considerably the number of CKPs.

The experimental results and comparisons support our claims, providing a reasonable evidence that an approach specifically built for Arabic, leveraging linguistic and syntactic knowledge can outperform western languages based approaches. Moreover, experimental results suggest also that moving the focus from candidate selection to candidate generation could provide a significant performance lift, especially in languages that have a rich morphology and syntax to leverage.

These results, however, are limited by the adopted datasets; as pointed out in [5], the Arabic language suffers, with respect to its number of speakers, from a tremendous lack of linguistic resources fit for NLP purposes. Our future work will be focused on coupling our CKP generation approach with a more advanced selection phase and on addressing the problem of linguistic resources shortage.

Acknowledgments

We'd like to thank Mahmoud Nabil, Mona Hedaya and Prof. Samahaa El-Beltagy for kindly sharing datasets and insights on their systems with us. Deepest appreciation to AbdulMouty Ahmad, Heshmat Taha, Muhammad AlHawal, and Asaad Abdurrahman for their enlightening and guidance.

References

[1] S. R. El-Beltagy and A. Rafea, "Kp-miner: A keyphrase extraction system for english and arabic documents," *Information Systems*, vol. 34, no. 1, pp. 132–144, 2009.

[2] T. El-Shishtawy and A. Al-Sammak, "Arabic keyphrase extraction using linguistic knowledge and machine learning techniques," in *Proceedings of the Second International Conference on Arabic Language Resources and Tools*. The MEDAR Consortium, 2009, pp. 1–8.

[3] R. Duwairi and M. Hedaya, "Automatic keyphrase extraction for arabic news documents based on kea system," *Journal of Intelligent and Fuzzy Systems*, vol. 30, no. 4, pp. 2101–2110, 2016.

[4] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics (ACL), 2003, pp. 216–223.

[5] A. Awajan, "Keyword extraction from arabic documents using term equivalence classes," *TALIP*, vol. 14, no. 2, p. 7, 2015.

[6] A. Farghaly and K. Shaalan, "Arabic natural language processing: Challenges and solutions," *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 8, no. 4, p. 14, 2009.

[7] N. Y. Habash, "Introduction to arabic natural language processing," *Synthesis Lectures on Human Language Technologies*, vol. 3, no. 1, pp. 1–187, 2010.

[8] W. Monroe, S. Green, and C. D. Manning, "Word segmentation of informal arabic with domain adaptation," in *Proceedings of the 52nd Annual Meeting of the ACL*. ACL, 2014, pp. 206–211.

[9] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proceedings of the 2003 Conference of the North American Chapter of the ACL on Human Language Technology-Volume 1*. ACL, 2003, pp. 173–180.

[10] M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki, "The penn arabic treebank: Building a large-scale annotated arabic corpus," in *Proceedings of NEMLAR Conference on Arabic Language Resources and Tools*, vol. 27, 2004, pp. 466–467.

[11] E. Al-Shammari and J. Lin, "A novel arabic lemmatization algorithm," in *Proceedings of the Second Workshop on Analytics for Noisy Unstructured Text Data*. ACM, 2008, pp. 113–118.

[12] T. El-Shishtawy and F. El-Ghannam, "Keyphrase based arabic summarizer (kpas)," in *Proceedings of 8th International Conference on Informatics and Systems (INFOS)*. IEEE, 2012, pp. 14–16.

[13] J. Hajic, O. Smrz, T. Buckwalter, and H. Jin, "Feature-based tagger of approximations of functional arabic morphology," in *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT)*, 2005, pp. 53–64.

[14] M. Nabil, A. Atiya, and M. Aly, "New approaches for extracting arabic keyphrases," in *Proceedings of First International Conference on Arabic Computational Linguistics (ACLing)*. IEEE, 2015, pp. 133–137.

[15] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning, "Kea: Practical automatic keyphrase extraction," in *Proceedings of the Fourth ACM Conference on Digital Libraries*. ACM, 1999, pp. 254–255.