International Journal of Computer Applications (0975 – 8887)
Volume 87 – No.7, February 2014

Orthogonal Processing for Measuring the Tonality of Egyptian Microblogs

Madeeh Nayer El-Gedawy
Computer Center
Institute of Public Administration (IPA) – Jeddah

ABSTRACT

Subjectivity and Sentiment Analysis (SSA) research in Arabic is still in its beginning phases regarding the research done in English on different granularities (sentence and document levels). In this paper, a simple system is proposed to perform sentiment analysis (or polarity detection) using an aggressive stemmer in the preprocessing phase followed by a Fuzzy classifier. The main focus of this paper is optimizing the preprocessing tasks for better tonality detection performance. Twitter is used as the data source because it is considered one of the hugest online dialectal Arabic microblogs repositories.

General Terms

Text Mining

Keywords

Sentiment – aggressive - stemmer – normalization – stops word removal – Fuzziers.

1. INTRODUCTION

The problem of SSA is defined as trying to identify opinion, opinion holder, polarity and sentence subjectivity [1]. In other words, extracting the tonality of the text. Different text granularities could be analyzed: sentence-level and document-level by using various algorithms and techniques some of them are supervised others are unsupervised. Polarity detection is of a special interest to the community because of the endless promising applications and benefits for companies and governments due to the very rich online data. Interestingly, social networked have changed the political map in Egypt, Tunis, and other Arab countries. For example, 30% of Egyptians uses social networks according to PEW research center, half of them their ages ranges from 18 to 29; this percentage is the same as for Japan according to the survey sample and results. Moreover, the Text and Retrieval Conference has introduced the microblog track in 2011. SSA applications range from ensuring customer satisfaction to predicting the next president in elections; also it could be used to recommend news, movies and friends to users.

For sentiment analysis, we need lexicon and tagged datasets to feed into the classifier. These resources are not sufficient in Arabic especially for the Egyptian dialect. This is why several researchers are annotating their datasets from facebook or twitter. Researchers that use sentiment orientation approach must depend on a well-tuned lexicon. Actually, MPQA [2] is the most known English lexicon used; it has 8000 words along with their polarity (positive, negative or neutral) and subjectivity. Also, there is ArabSenti [3] that has 4000 adjectives along with their polarity. A one approach used to enrich the lexicon is to use a graph reinforcement algorithm like Random Graph Walk to extend the lexicon using phrase tables by detecting extra mappings.

Arabic is a language of complex morphology, it has thousands of roots, and it's hard to detect the correct roots because of clitics (prefixes and suffixes) and infixes that decorate the word and may change the meaning. One big problem in sentiment analysis is detecting the negation as it reverses the sentiment polarity, the rules of negation in Modern Standard Arabic differ from Egyptian dialect that uses two rules: مش+ adj (مش فاهم), or ما+verb+ش (ما ينفعشى). The easiest way to deal with such complexities is to use n-grams. Moreover, many tweets and facebook comments are sarcastic. Egyptian dialect sentences are exposed to elongations such as: لووووول. The Egyptian dialect has its own stop word list that must be considered along with the MSA stop word list Taking into account the nontrivial amount of spelling mistakes and that Egyptian dialect words could be written in several ways like: ما اعرفشى –ما اعرفش and معلش–معلهش and نضر – نظر. Generally, dialects change the morphology and pronunciation. Many text mining tasks have been handled to improve the systems performance such as: colloquial to standard translation, elongation removal, spell checking, treating out of vocabulary words [4] [5].

There are three steps executed in the preprocessing: normalization, stemming, and stop word removal. Words are normalized to get a more coherent form, while stemming is meant to remove the inflections decorating the root or the stem, there are two approaches used for stemming: aggressive stemming and light stemming, aggressive stemming tries to reach the root of the word while the light stemming tries to find the fewest letters of the word that are sufficient to keep the word meaning, and at last we remove functional words that do not add any useful meaning to the analysis such as pronouns, auxiliary verbs, prepositions and determiners.[6][7] investigated the effect of orthogonal processing upon polarity detection. Yet, there are some useful tools used in the community such as Stanford segmenter, Sebawai and Tashaphyne for stemming.

Many features could be used and fed into the classifiers such as: POS tags such as nouns, phrasal verbs, transitive, intransitive verbs and specific features of the tweets such as emoticons and retweets. For sure, the most prominent features are positive, negative adjectives, stems along with their polarity [8]. Interestingly, the majority of Arabic tweets are negative [9], thus all work done by building equal amount of positive and negative tweets will be misleading, specially, if the classifier is a Naïve Bayes. Assuming that tweets broadcasted as news are neutral is a wrong assumption [10] too. Bigrams and trigrams (stem context) give better results than of unigrams because they detect negation and grasp the polarity more accurately.

Sentiment analysis is conducted either by various machine learning techniques or using unsupervised approach. If machine learning is used, one have to get a dataset exemplars tagged as positive or negative, then these exemplars are fed to the classifiers. If the analysis is conducted through an unsupervised approach, one has to build a lexicon of positive and negative terms along with their predefined polarity. Afterwards, one can detect the sentence tonality by summing the polarities of all sentence words that are listed in the lexicon.

2. RELATED WORK

Preprocessing begins with tokenization and normalization. The Arabic alphabet consists of 28 letters. However, some extra characters are being used; they correspond to different forms of some specific letters of the alphabet. These extra letters contain: various forms of alef, their presence depends on the morphology and context of the word, but they are usually exchanged by mistake. AlefMaksoura (ى) which is always confused with (ي). Your marbouta (ة) coincides with (ه). Different forms of hamza, depending on the word POS. For instance: ماؤه - ماءه - مائه, are all the same word 'water' but with different part of speech. Characters such as: Kashida and diacritics need to be removed. So, in a nutshell, in the normalization step, we deal with tashkeel, tanween, hamza, alef, lamalef, yeh and heh. Amira [11] could be used for this normalization step, also there is a normalize available for download in Ruby.

Next, is the stemming step. Actually, Arabic has about 10,000 roots, nearly half of them are commonly used. A root has a generative nature, thus it can generate dozens of different meanings corresponding to its different lexical forms. Stemming leads to two problems: inflection (attaching letters to the word without changing the meaning: "جيد - جيدة") and derivation (attaching letters to the wordchange themeaning : " جيد–جيد(رقبة) - جياد "). There are two kinds of stemmers: aggressive stemmers and light stemmers. Root extractors are aggressive stemmer that attempt to find the word root, where many words of different meanings can be conflated to the same root. For example: " فلنولينك قبلة ترضاها"it is obvious that the word " فلنولينك"has the root "ولى" which could mean a very pious man, or "ولاية"a state or trusteeship, or the colloquial word "ولية"which means a woman, or as a verb it could be to give or make someone a ruler on a country. Thus, we encounter the problem of overstemming where the target meaning could be lost from the original text . On the other hand, light stemming try to find the fewest letters that preserve the meaning, but sometimes it fails due to affixes and irregularities.

The Khoja [12] stemmer is a very well-known aggressive stemmer; it removes all diacritics, determiners, punctuation marks, the conjunction prefix 'waw' and numbers. All words are then checked against its exhaustive list of prefixes and suffixes, if there is a match, the longest match will be cut, at last the word is compared to some patterns, if there is a match, and then the root is determined. Another well-known stemmer is light10 [13], which is a light stemmer that is fatherly extended by [14], this specific extension could identify broken plurals and generate the stems in their singular forms. Actually, the same idea was previously investigated by [15] as an extension to Khoja aggressive stemmer.[16] introduced an Arabic stemmer of 97.1% accuracy, whereas [11] has reached 99.2% accuracy by training a SVM classifier trained on Arabic Treebank.

Al-Shalabi [17] has devised a very straightforward root extractor that depends heavily on heuristics. We will discuss this approach in detail as it is used in this preprocessing implementation; the main advantage of this approach is that it has a simple implementation which was coded in C#. Firstly, number of letters in the word is checked whether it is less than or equal to 3, then the word will not be stemmed, else the following simple steps are followed:

1- For each letter in the term (from right to left) apply weight and rank values according to Tables I and II.
2- Measure the product of the rank and weight for each letter.
3- Keep only the letters with the smallest first three values.

The rank of a word is calculated differently when its number of letters is odd from that when it is even (table 1). The weights of letters are given values for letters categorized into groups as shown in table 2.

Table 1. Letters ranks according to positions

Letter position from right	Rank (if word length is even)	Rank (if word length is odd)
1	N	N
2	N-1	N-1
3	N-2	N-2
..	..	..
[N/2]	[N/2]+1	[N/2]
[N/2]+1	[N/2]+1 - 0.5	[N/2]+1 - 0.5
[N/2]+2	[N/2]+2 - 0.5	[N/2]+2 - 0.5
..	..	..
N	N- 0.5	N - 1.5

Table 2. Weights of letter groups

Arabic Letters	Weight
ا ة	5
ي ئ	3.5
ت ى و	3
أ إ م ن	2
ل س ه	1
Rest of the Arabic Alphabet	0

Al-Shalabi did not clarify the theoretical foundation these ranking and weighting; only these classes and values were chosen after deep experimentation. In table 3, we show how the word "محمد" is stemmed using these simple steps.

Table 3. Shows how the word "محمد" is stemmed using Al-Shalabi

د	م	ح	م	Letters
4	3	2	1	Order
0	2	0	2	Weight
3.5	2.5	3	4	Rank
0	5	0	8	Product
حمد				Root

Sometimes Al-Shalabi confronts some discrepancies such as stemming the word "التعليمات" as shown in table 4, for this reason we have used an adjusted version of Al-Shalabi where the letter "ل" has the weight "3", the letter "س" has the weight "2", and the letter "ه" has the weight "5". Table 5 shows the stemming of the word "التعليمات" using the adjusted approach.

Table 4. Shows how the word "التعليمات" is stemmed using Al-Shalabi

ت	ا	م	ي	ل	ع	ت	ل	ا	Letters
9	8	7	6	5	4	3	2	1	Order
3	5	2	3.5	1	0	3	1	5	Weight
7.5	6.5	5.5	4.5	5	6	7	8	9	Rank
22.5	32.5	11	15.75	5	0	21	8	45	Product
لعل									Root

Table 5. Shows how the word "التعليمات" is stemmed using adjusted Al-Shalabi

ت	ا	م	ي	ل	ع	ت	ل	ا	Letters
9	8	7	6	5	4	3	2	1	Order
3	5	2	3.5	2	0	3	2	5	Weight
7.5	6.5	5.	4.5	5	6	7	8	9	Rank
22.5	32.5	11	15.75	10	0	21	16	45	Product
علم									Root

Last step is stop word removal; there are many words that could exist in a document, it is necessary to filter the noise out of the text to keep only the important terms. Stop words are examples of noise in the data because they do not contribute to the meaning of documents and they don't have any added value for the text. Most of these words are not relevant to the classification task and can be removed without affecting the performance of classification, and can even lead to an improvement due to noise reduction, but it is not recommended that the stop list gets too much extended [18].

Many text classifiers take out stop words; this deletion could be very aggressive to the level that 90 percent of all the terms are omitted [19]. The elimination of stop words could cut the size of the corpus down to 25% [19]. Stop words could also be specific to different domains [20], for example, the word "خيط"may be a stop word in a text addressing textile industry, but certainly it will not be a stop word in this verse: " حتى يلج الجمل في سم الخياط". Truthfully there is no one specific list that all researchers go with; there are some lists available online such as [21]. Anyway, some scholars restrict these lists to functional words such as prepositions and determiners; others add the most popular words such as: عايز– تعمل – تعرف. [9] Built a stop word list for the Egyptian dialect –which is one of the six dominant dialects in the Arab world [22]- out of 20000 tweets. [23] used the 162 MSA stop words and added other 90 dialectal words for terms like: اللى– مش.

3. SYSTEM ARCHITECTURE

The system works in sequential three steps: crawling microblogs to form the corpus and annotating the dataset, preprocessing, dealing with dialects, elongations and specifying the features vector, and then running the classifier.

3.1. Crawling and annotating the dataset

Twitter4j has been used to crawl Twitter, we collected 10000 tweets from June to December 2013, then we selected a sample of 1500 tweets which are annotated by hand either as positive or negative; only positive and negative tweets are considered, neutral and sarcastic tweets are neglected. Two individuals annotated the same 1500 exemplars; they agreed on nearly 91% of them, the other 9% has been neglected. The final dataset composed of 1200 tweets, 800 of them are negative and 400 are positive, not equal quotas. Every tweet is covering the sentiment of its writer towards only one topic.

3.2. Preprocessing

Firstly, for normalization we apply the following rules as listed in next table (table 6)

Table 6. Normalization rules

Rule	Example
Tashkeel	ًٌَُ ◌
Tatweel	Aaaaaaaaalah> Allah
Hamza	ء -> ء ئ ؤ
Alef	ا<- ا أ إ
lamalef	لإ -> لا لأ لآ
yeh	ى -> ي ي
heh	ة -> ة or ه

For dealing with elongations, instead of automatically deleting repeated words, we implemented a better algorithm developed by [24] which works as follows: if the word exists in the dictionary, then it is left as it is, else a compression version of the word is generated even if there is no repetitions, if the word exists in the compressed list, then it is replaced by the most frequent surface form even it this form has more repeated letters.

The Egyptian dialect needs special handling before stemming, thus, we run the terms through two components: the first component is a small dictionary manually filled with Egyptian dialect words, actually, we built a short list of words containing 270 words that should not be stemmed, we manually gathered these words by looking up 1000 tweets. Obviously this procedure is not sufficient, so we check the terms against an algorithm developed by [25] that attempts to transfer some Egyptian dialect words into MSA words by applying some rules that make multiple lexicon lookups as the mapping involves more than one morpheme. Table 7 shows some these relations.

Table 7. Mapping Egyptian dialect words into MSA

Egyptian Dialect	MSA	Mapping
ايد	يد	Vowel replacement
احنا	نحن	Pronoun distortion
اتبل	ابتل	Swapping letters
تمطع	تمطى	Replacing characters

Afterwards, the modified Al-Shalabi aggressive stemming algorithm is taking place as explained. For features vector, we worked with unigrams, bigrams, and trigrams. Unigrams are the simplest features for the data, bigrams and trigrams are better in grasping the negation. Out of the 1200 tweets, we created a dictionary for all the candidate features along with their frequencies. In testing, for each tweet, if any of these candidate features is present, then this candidate feature frequency is fetched from the dictionary and it is placed in the feature vector representing this tweet.

Lastly, An entropy based approach has been adjusted for creating an Arabic stop word list for this WSD system as explained by [26][27]. The dataset used for extracting the stop word list is a sample of the NEWSWIRE corpus that contains 17,000 articles and another dataset for facebook comments that contain 2000 exemplars to get both MSA and Egyptian dialect stop words.

Step 1: Word frequency is the number of times a word appears in a document. The list is sorted in descending order of frequency.

Step 2: we measure the likelihood Li,j of the term wj in document Di:

$$\text{Li,j} = \frac{\text{frequency in the document Di}}{\text{the total number of words in document Di}}$$

Then we calculate entropy that measures the information value of the word w_j:

$$\text{H}(\text{w}_\text{j}) = \sum \text{Li, j} \, * \, \log(1/\, \text{Li, j})$$

3.3. The Classifiers

Although the extensive use of Naïve Bayes and SVMs in many text classification problems especially because of the ease of implementations in software and toolkits such as: NLTK, RapidMiner and Weka, , we preferred to use two Fuzziers that we have previously suggested [28], they were originally developed to disambiguate words, but here we use them to this simple text classification task; the first classifier is a Sigmoid function fuzzier, where the relationship between the terms in each microblog and class (positive or negative) can be expressed as a degree of memberships that formulate a fuzzy set for this class. Thus, each sense is expressed by a fuzzy set, as follows:

$$\text{FS}\ (\text{S}_\text{y}) = \mu\ (\text{w1, Spos}) \ + \ \mu\ (\text{w2, Spos}) \ + \ \ldots\ldots + \mu\ (\text{wk, Spos})$$

Where:

- FS (Sy): the fuzzy set of class 'pos'.
- w1: the first word in the microblog.
- k: number of words in the microblog.
- μ (w1): the weight or the membership that expresses the degree of truthfulness.
- μ (w1,spos): a fuzzy logic terminology; it means: [how much the word 'w1' should be allocated to the class 'pos' for the word 'w']. it is calculated by this formula [29]:

$$\mu(\text{w1,s}_\text{pos}) = 0.3 \ + \ 0.7 \ \frac{1}{1+ e^{-2\,(TF\ for\ this\ word\ in\ this\ class)}}$$

The second classifier is a Jaccard-based where the relationship between a term't' and a class'pos' can be formulated as:

$$\mu\ (\text{t}, \text{s}) = \frac{\text{total number of term frequency in this class}}{\text{total number of term frequency in all classes}}$$

Where:

- μ(t,pos): the truthfulness degree that the term 't' belongs to a specific class 'pos'.
- (TF): the term frequency of the context word in the training set.
- And for testing we apply this formula:

$$sim(test\ exemplar, class\ 'pos') = \frac{\sum[\boldsymbol{\mu}\ (\mathbf{t}, \mathbf{pos})]\ \text{conjunction [TF in test set]}}{\sum[\boldsymbol{\mu}\ (\mathbf{t}, \mathbf{pos})]\ \text{disjunction [TF in test set]}}$$

Where:

- TF: denotes the term frequency in the test example.
- Conjunction and disjunction: operators will be substituted by four functions: Algebraic, Minimum-Maximum, Hamacher, and Einstein as shown in next table (table 8)

Table 8. Conjunction and Disjunction operators

Method	Conjunction (a,b)	Disjunction (a,b)
Algebraic product	$a * b$	$a + b - a * b$
Minimum-Maximum	Min{a,b}	Max(a,b)
Hamacher Product	$\frac{a * b}{a + b - a * b}$	$\frac{a + b - 2 * a * b}{1 - a * b}$
Einstein Product	$\frac{a * b}{2 - [a + b - a * b]}$	$\frac{a * b}{1 + a * b}$

As Einstein product yielded the best results, then it will be sufficient as a proxy for this method.

4. EXPERIMENTATIONS

To examine the performance of these proposed orthogonal processing steps, two experiments are executed: 1) using unpreprocessed microblogs, 2) after applying all the preprocessing tasks, the results are shown in tables 9 and 10. The Fuzziers are first trained using the frequency of the unigrams only; then they were trained using both unigrams and bigrams and finally they were trained using all unigrams, bigrams and trigrams. The results were as follows using 10-fold cross validation with 90/10 training/test splits:

Table 9. Sentiment Fuzziers results without preprocessing

	Sigmoid-based & Jaccard-based Fuzziers		
	Precision	Recall	F-Measure
Unigrams(Sigmoid)	0.75	0.74	0.74
Unigrams(Jaccard)	0.72	0.72	0.72
Unigrams+Bigrams(Sigmoid)	0.74	0.74	0.74
Unigrams+Bigrams(Jaccard)	0.72	0.72	0.72
Unigrams+Bigrams+Trigrams (Sigmoid)	0.74	0.73	0.73
Unigrams+Bigrams+Trigrams (Jaccard)	0.72	0.72	0.72

Table 10. Sentiment Fuzziers results after finishing all preprocessing tasks

	Sigmoid-based & Jaccard-based Fuzziers		
	Precision	Recall	F-Measure
Unigrams (Sigmoid)	0.79	0.79	0.79
Unigrams (Jaccard)	0.78	0.77	0.77
Unigrams+Bigrams (Sigmoid)	0.81	0.8	0.8
Unigrams+Bigrams (Jaccard)	0.8	0.8	0.8
Unigrams+Bigrams+Trigrams (Sigmoid)	0.81	0.81	0.81
Unigrams+Bigrams+Trigrams (Jaccard)	0.8	0.8	0.8

5. CONCLUSION AND FUTURE WORK

It is obvious that the results obtained after orthogonal preprocessing are better than those obtained without applying them. The best raise in performance is 7%. From the two previous tables, it can be easily noticed that unigrams yield the best results if no preprocessing is taking place as the F-measure for the Sigmoid fuzzy classifier is 74%, whereas the combination of unigrams, bigrams, and trigrams yield the best results when applying all preprocessing steps, as the F-measure for the Sigmoid fuzzy classifier is 81%. It is also worth mentioning that the Simgoid Fuzzier yields better results than those obtained by applying the Jaccard-based Fuzzier; the difference is almost 1%.

For future work, the performance of light and aggressive stemmers in the task of sentiment analysis will be examined, as it is important to handle sarcastic microblogs too as they represent a reasonable percentage of all comments and tweets. The author would like to compare the results with a sentiment orientation approach and maybe a hybrid of the two approaches could be achieved. Moreover, negation can be handled using a more accurate approaches.

6. REFERENCES

[1] Bing Liu. Sentiment analysis and subjectivity (2010). Handbook of Natural Langauge Processing, pages 627-666.

[2] Johansson, Richard (2013). "Relational Features in Fine-Grained Opinion Analysis". Computational linguistics, Association for Computational Linguistics (0891-2017), 39 (3), page 473.

[3] Muhammad Abdul-Mageed, Mona Diab, and Mohammed Korayem (2011). Subjectivity and sentiment analysis of modern standard arabic. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 587–591.

[4] Hasan Muaidi and Rasha Al-tarawneh (2012). Towards Arabic Spell-Checker Based on N-Grams Scores. International Journal of Computer Applications 53(3), pages:12-16,. Published by Foundation of Computer Science, New York, USA.

[5] Vipperla, Ravichander,Frankel, Joe,Kin (2012). Direct posterior confidence for out-of-vocabulary spoken term detection, ACM Transactions on Information Systems (TOIS), volume 30, number 3.

[6] emalatha, G. P Saradhi Varma ,A.Govardhan (2012). Preprocessing the Informal Text for efficient Sentiment Analysis, Preprocessing the Informal Text for efficient Sentiment Analysis, volume 1, number2.

[7] Emma Haddia,Xiaohui Liua,Yong Shib (2013). The Role of Text Pre-processing in Sentiment Analysis, Procedia Computer Science, Volume 17, pages 26–32.

[8] Isa Maks, Piek Vossen (2012). A lexicon model for deep sentiment analysis and opinion mining applications, Decision Support Systems, Volume 53, number 4, pages 680-688.

[9] Ahmed Mourad, Kareem Darwish (2013). Subjectivity and Sentiment Analysis of Modern Standard Arabic and Arabic Microblogs, Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pages 55–64, Atlanta, Georgia.

[10] Alexander Pak and Patrick Paroubek (2010). Twitter as a corpus for sentiment analysis and opinion mining, In Proceedings of LREC, volume 2010.

[11] M. Diab. 2009. Second Generation Tools (AMIRA 2.0): Fast and Robust Tokenization, POS tagging, and Base Phrase Chunking. Proceedings of the Second International Conference on Arabic Language Resources and Tools, 2009.

[12] S. Khoja, R. Garside (1999). Stemming Arabic text, Tech. rep. Computing Department, Lancaster University, Lancaster, U.K.

[13]Mohammed A.Otair (2013). Comparative analysis of Arabic stemming algorithms, International Journal of Managing Information Technology, volume 5.

[14] Samhaa R. El-Beltagy, Ahmed Rafea (2011). An accuracy-enhanced light stemmer for arabic text , ACMTransactions on Speech and Language Processing (TSLP) , Volume 7 number 2 .

[15] Goweder, A., Poesio, M., De Roeck, A., Reynolds, J.: Identifying broken plurals in unvowelised Arabic text. In: EMNLP 2004, Barcelona, Spain (2004).

[16] K. Darwish, H. Hassan, O. Emam (2005).Examining the Effect of Improved Context Sensitive Morphology on Arabic Information Retrieval. CASL workshop in ACL.

[17] R. AI-Shalabi, G. Kannan, and H. AI-Serhan (2003). "New Approach for extracting Arabic roots". In Proc of 2003 International Arab conference on Information Technology, Alexandria, Pages: 42-59.

[18] Zhou Yao, Cao Ze-wen (2011). "Research on the Construction and Filter Method of Stop-word List in Text Preprocessing". Proceedings of the 2011 Fourth International Conference on Intelligent Computation Technology and Automation, Volume 1.

[19] Ronen Feldman, James Sanger (2006). "Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data". Cambridge University Press, NY, USA.

[20] Mehdi Khosrow-Pour (2008). "Encyclopedia of Information Science and Technology, 2 edition". Information Science Reference - Imprint of: IGI Publishing Hershey, PA.

[21] Amira Shoukry (2013). Arabic Sentence-level sentiment analysis, A Thesis Submitted to The Department of Computer Science and Engineering, AUC, Cairo, Egypt.

[22] Mona Diab, Nezar Habash (2009). Arabic Dialect processing. MEDAR 2009, Cairo, Egypt.

[23] Amira Shoukry, Ahmed Rafea (2012). Preprocessing Egyptian Dialect Tweets for Sentiment Mining, In proceeding of: Fourth Workshop on Computational Approaches to Arabic, AMTA.

[24] Karim Darwish, Walid Magdy, Ahmed Mourad (2012). Language processing for arabic microblog retrieval, in proceedings of the 21st ACM international conference on Information and knowledge management, pages 2427-2430 ACM New York, USA.

[25] Khaled Shaalan, Hitham Bakr, Ibrahim Ziedan (2007). Transferring Egyptian Colloquial Dialect into Modern Standard Arabic, in international conference on recent advances in Natural Language Processing (RANLP), pages 525-529.

[26] Feng Zou, Fu Lee Wang, Xiaotie Deng, Song Han, Lu Sheng Wang (2006). "Automatic construction of Chinese stop word list". Proceedings of the 5th WSEAS international conference on applied computer science, Pages: 1009-1014.

[27] A. Alajmi, E. M. Saad, R. R. Darwish (2012). “Toward an ARABIC Stop-Words List Generation”. International Journal of Computer Applications (0975 – 8887), Volume 46, Number 8.

[28] Madeeh Nayer El-gedawy (2013). Using Fuzzifiers to solve Word Sense Ambiguation in Arabic Language, International Journal of Computer Applications 79(2):1-8, New York, USA.

[29] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery (2007). "Numerical Recipes 3rd Edition: The Art of Scientific Computing, 3 edition". Cambridge University Press, New York, USA.