2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Tagging Documents using Neural Networks based on Local Word Features

Arnulfo P. Azcarraga, Paolo Tensuan
College of Computer Studies
De La Salle University,
Manila, Philippines
arnie.azcarraga/paolo.tensuan@delasalle.ph

Rudy Setiono
Department of Information Systems
School of Computing
Singapore
rudys@comp.nus.edu.sg

***Abstract*—Keywords and key-phrases that concisely represent text documents are integral to many knowledge management and text information retrieval systems, as well as digital libraries in general. Not all text documents, however, are annotated with good keywords; and the quality of these keywords is often dependent on a tedious, sometimes manual, extraction and tagging process. To automatically extract high quality keywords without the need for a semantic analysis of the document, it is shown that artificial neural networks (ANN) can be trained to only consider in-document word features such as word frequency, word distribution in document, use of word in special parts of the document, and use of word formatting features (i.e. bold-faced, italicized, large-font size). Results show that purely local features are adequate in determining whether a word in a document is a keyword or not. Classification performance yields a *G mean* of a least 0.83, and *weighted f-measure* of 0.96 for both keywords and non-keywords. Precision for keywords alone, however, is not as high. To understand the basis for classifying keywords, C4.5 is used to extract rules from the ANN. The extracted rules from C4.5, in the form of a decision tree, show the relative importance of the different document features that were extracted.**

Keywords—keyword extraction; document tagging; feature selection; artificial neural network; scientific documents

I. INTRODUCTION

In this digital age, huge amounts of information in the form of images, video-clips, and most of all, text documents have made it more and more difficult for human users to sift through the relevant and irrelevant information. Keywords and key phrases, as concise and meaningful representation of text documents, have thus become integral to many knowledge management and text information retrieval systems, as well as digital libraries in general [1][2][3][4]. Web search services are likewise aided enormously when the most important words of every text-document are available. Keywords have also become significant in e-commerce, specifically in providing contextual advertisements to online content, as well as for online news, and other highly specialized digital libraries [5][6][7][8].

Automatic keyword generation, selection, or extraction have thus become a very important field of research in Computer Science and Computational Linguistics [9][10][11]. The field comes under other names, each with slightly different emphasis, including *document tagging* and *annotation*, as well as the more general problem of selecting bigrams, and even entire phrases, more aptly referred to as *key phrase selection*.

One approach to tagging documents is to simulate the way human experts go about in annotating documents. This would involve deep semantic analysis, and would include understanding the nuances of words, understanding figures of speech, idiomatic expressions, and the like. Such analyses would also typically involve some syntactic analysis of each sentence, extraction of named-entities and parts-of-speech (POS), and semantic analysis of these sentence items in order to understand the meaning of sentences, paragraphs, sections, and entire documents.

Another approach, often related closely with Natural Language Processing (NLP) techniques, is through the use of carefully crafted rules for keyword extraction, in the form of expert systems[12][13][14][15][16]. This approach may be effective in specialized domains, and closed archives, but they typically would require some manual construction of a set of inference rules on very particular linguistic styles and domains of knowledge. Such an approach would have difficulty in scaling up to very large document sets because of the tedious effort needed for each domain and for each linguistic style, or manner of writing.

Yet another approach is the use of statistics-based methods, such as Bayesian networks, k-nearest-neighbor algorithms, and Expectation Maximization. Other approaches that are statistical and computational in nature, use data-driven machine learning algorithms to perform the task of distinguishing keywords from non-keywords would include Genetic Algorithms, Support Vector Machines, Decision Trees, Self-Organizing Maps, and Artificial Neural Networks [17][18][19][20][21][22][23].

These methods may yield quite accurate keywords, but for these methods to be scalable, they should not rely on features that are difficult and tedious to extract, such as when ITF and IDF features are used. The inverse document frequency (IDF), for example, would need the frequency counts of all words in all the documents in the corpus before it can be computed. Hence, methods that rely on such features can be troublesome when employed to handle rapidly growing document sets, such as the documents found on the Web.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

The approach described in this paper does not attempt to do any level of semantic analysis of the individual sentences, nor any kind of natural language processing that would seek to understand what is being discussed. In particular, our approach is to use *artificial neural networks (ANN)* that are fed with word features based on frequency, position, usage, and format - and just on the basis of these features, the ANN learns to distinguish important words (keywords) from non-important words. None of these features would allude to any form of semantic analysis, nor any look-up of some online thesaurus or dictionary to check for the meaning of words. These features are very fast to compute and extract from each text document.

ANNs have been proven to be universal approximators [24] which are theoretically able to learn any, possibly non-linear, relation between a set of input features to some set of output items, given the proper features and an adequate data sample. Indeed, ANNs have been used for keyword extraction [25][26][27][28] and the work reported in this paper builds on past work reported in [25].

The rest of the paper is organized as follows: section II describes the dataset; section III describes in some detail the various word features used and how they are extracted from the individual documents; section IV describes the five experiments that were performed; section V discusses the results; and finally section VI concludes the paper.

II. DATASET

The same IEEE documents previously used in [25] are used for all the experiments reported here. The dataset is composed of IEEE journal articles on different scientific topics. Each journal document would typically have a title, an abstract, some key index terms (i.e. user supplied keywords), the text body, and a reference list. The IEEE dataset originally had 300 documents, with 150 documents that were randomly chosen for training and 150 randomly chosen documents for testing. A number of the training documents did not have an abstract or did not have a list of keywords. So, after filtering out non-compliant documents, a total of 270 documents were used in the final dataset, with only 120 training documents left.

Every document is pre-processed, going through standard text processing procedures prior to feature extraction. These pre-processing steps include tokenization, stop word removal, and stemming [33]. Each document is then represented as a sequential list of stemmed words which are then transformed into individual word feature vectors, also following standard text processing procedures [34][35].

In total, the processed documents yielded 427,185 word feature vectors. The training set has 69,190 word vectors, of which only 1.7% or 1,157 are keywords. The test set has 80,055 word vectors, of which 2.2% or 1,730 are keywords. A validation set, composed of 6,688 word vectors, of which 1.7% or 116 were keywords, were randomly selected from the training set.

Due to the heavily imbalanced nature of the dataset, where only around 2% of the words are labeled as important words (keywords), standard over-sampling was done on the training and validation sets. Over-sampling is a method of repeating word vectors from the under-represented class(es) so that the number of sample data for the different classes would even out.

III. WORD FEATURES

Each unique word in a document yields a word feature vector. These feature vectors include the document identifier, the word itself, and the selected word features for the dataset. We concentrate just on features involving the frequency, position, usage and format of a word in a document – or a total of 14 word-features. All these features are very easily computable given the word list (and their XHTML tags) that represents each document. The features are purely computational and can be extracted without resorting to any form of semantic, and not even syntactic, analysis of the sentences, paragraphs and sections of the document.

A. Word Frequency Feature

Word frequency is the number of times, normalized, that a term appears in the document. The frequency (***count***) of a word can be extracted by the following formula, where $n_{i,j}$ refers to the frequency of term t_i in document d_j:

$$\mathrm{tf}_{i,j} = \mathrm{n}_{i,j} / \sum_k n_{k,j} \quad (1)$$

B. Position Features

Some writing styles recommend that the first sentence of the paragraph should give the main thought, while other sentences in the paragraph would only support the opening sentence. In this kind of writing style, the words that appear somewhere at the start of a paragraph can indicate that the word is important. As such, the average position of a given word in the different paragraphs (***pos-in-paragraph)*** might be a useful feature to extract. Given the number of words in the paragraph p as $\mathrm{N_P}$ and k as the position of the word in the paragraph, a term's position in a paragraph is computed as follows:

$$\mathrm{pos(paragraph)} = \mathrm{k/N_P} \quad (2)$$

The average position is computed if the term appears more than once in the document.

There is a similar feature extracted to represent the average position of the word in all the sentences (***pos-in-sentence***) of the document. Given the number of words in sentence s as $\mathrm{N_S}$ and k as the position of the word in that sentence, a term's position in a sentence is computed as

$$\mathrm{pos(sentence)} = \mathrm{k/N_S} \quad (3)$$

The average position is also computed if the term appears more than once among the different sentences of the document.

C. Frequency in specific segments of document

Each document is divided into five segments, based purely on the number of words that appear in the document (i.e. with no regard for the logical sections and subsections of the document). The frequency of each word is then computed in each of segments 1 to 5. We denote as $f_{i,j}(s)$ the frequency of the word i in the s^{th} segment of document j. The frequency computed below gives an indication of where from among the five segments is the given word most frequently appears. In that segment, $f_{i,j}(s)$ is 1. As for the other segments s, the value of $f_{i,j}(s)$ would be anywhere from 0 to 1, representing the ratio of the frequency of the word in a given segment relative to its frequency where it appears most often, as defined in (4).

$$f_{i,j}(s) = tf_{i,j}(s)/[\max\{tf_{i,j}(k)\}, k = 1,2, \ldots, 5] \quad (4)$$

The special segment-specific frequency features computed above as $f_{i,j}(s)$, s = 1,2,..5, are referred to as ***pos1, pos2, … pos5***. These features are extracted because words that frequently appear in certain segments of the document (e.g. ***pos1*** and ***pos5,*** the first and last segments) are predicted to be the more important words.

D. Word Usage Features

The word usage features are Boolean values, with a value of 1 if the word has been used at least once as part of at least one section heading (***in-heading***), as part of the abstract (***in-abstract***), or as part of any of the Figure or Table captions (***in-caption***).

E. Word Formatting Features

Similarly, the word formatting features are Boolean values, with a value of 1 if the word has been italicized at least once in any part of the document other than the title, abstract and section headers (***italicized***), bold faced, (***bold faced***), or written using a font size that is larger than the most common font size (***is-big***).

IV. Experimental Set-Up

Aside from the 14 features discussed earlier, each word feature vector carries a *label* – an indicator as to whether the word is an important word (keyword) or not. Having no access to an explicit list of "important words" per document, we had to resort to proxy labels. *A word is labelled as a keyword if appears in the document title or in the list of keywords found right after the abstract*. Indeed, these are the words, chosen by the document author/s, which are probably important.

Aware that such proxy labels are inadequate in that some title words are not keywords (although stop words in the title would have been removed), and some keywords do not appear in the title nor keyword list, we are constrained to use this as basis for tagging words as keyword or non-keyword. It was argued in the earlier study [25] that in the absence of a more authoritative dataset that has manually selected keywords (by experts), it is justifiable to use title words and words in a keyword list as basis for proxy labels. This is preferred over allowing researchers themselves to assign the desired labels per document.

Once all the word vectors have been processed, feature data values are normalized and the resulting word feature vectors are used to train an ANN, with *backpropagation* as the machine learning method used. All the experiments were run using *RapidMiner* [36]. The backpropagation network has 14 input nodes, one hidden layer consisting of nine hidden nodes, and two output nodes. One output node is trained to classify keywords, while the other is trained to classify non-keywords.

A threshold is set for the output nodes. Following the standard feed-forward activation of the nodes in multi-layered perceptrons, every time an output node's output value exceeds the threshold, the word is classified as keyword (or non-keyword), depending on which of the two nodes fires.

Experiment I provides the baseline information for assessing how much gain in performance would result from adding format features during training. Note that in the 2012 study [25], format features (italicized, bold-faced, and large font size) were not included. Experiment II used all 14 features, including the format features. As for Experiment III, the same 14 features as in Experiment II were used, except that there is a validation set that is used to determine the output threshold for the trained networks (referred to as "calibrated neural network").

V. Results and Discussion

The results from Experiments I and II show the skewed classification performance of the trained neural network. The trained classifiers perform better in recalling the non-keywords, as around 90% of the 78,325 non-keywords in the test set have been correctly classified. The trained classifier is also quite accurate (precise) whenever it predicts a word as a non-keyword. Of the more than 70,000 words predicted as non-keywords, indeed more than 99% are found to be truly non-keywords.

The confusion matrices for experiments I, II, and III are shown in Table I, while the *recall* and *precision* rates as well as the other summative performance rates are provided in Table II. From Experiment I, when only frequency, position, and usage features are used, accuracy (*recall*) for keywords is at 0.77. The figure is higher at 0.89 for non-keywords. Precision is quite low for the keywords, at only 0.13, but is very high at 0.99 among non-keywords. Slightly better results are obtained in Experiment II, attributable to the inclusion of format features for training. *Recall* for keywords are steady at around 0.76, and at 0.91 for non-keywords. As to *precision*, although it has improved from the 0.13 in Experiment I, it is still quite low for the keywords at only 0.16, but remains very high at 0.99 among non-keywords.

The same excellent *precision* results cannot be said for the case of keywords. Although, of the 1,730 keywords in the test set, more than 1,300 have been correctly classified as keywords, and thus a *recall* of about 76-77% for Experiments I and II, the *precision* for keyword prediction is consistently quite low. For Experiment II, of the more

than 8,516 words that were predicted to be keywords, only 1,323 are indeed keywords, leaving many words misclassified – and thus a *precision* of only about 16%.

This observation is all the more important to point out because the study is mainly concerned with the extraction of keywords, and therefore the assessment of the *recall* and *precision* would normally be focused on the keywords, and not the non-keywords.

TABLE I. CONFUSION MATRICES (CONTINGENCY TABLES) FOR THE THREE EXPERIMENTS, DENOTED BY THE INPUT FEATURES USED: FREQUENCY (**FREQ**), POSITION (**P**), USAGE (**U**) AND FORMATTING (**F**)

		Classified as					
Candidate	*freq*	I (Freq, P, U)		II (Freq, P, U, F)		III (Freq, P, U, F)	
words		K	NK	K	NK	K	NK
Keywords (K)	*1,730*	1,331	399	1,323	407	1,003	727
Non-Keywords (NK)	*78,325*	8,538	69,787	7,193	71,132	3,207	75,118
total	*80,055*	9,869	70,186	8,516	71,539	4,210	75,845

TABLE II. SUMMARY OF PERFORMANCE MEASURES FOR THE THREE EXPERIMENTS.

	EXPERIMENTS		
Performance Measures	I	II	III (calibrated)
	Freq, P, U	Freq, P, U, F	Freq, P, U, F
recall (K)	0.77	0.76	0.58
recall (Non-K)	0.89	0.91	0.96
precision (K)	0.13	0.16	0.24
precision (Non-K)	0.99	0.99	0.99
f-measure (K)	0.23	0.26	0.34
f-measure (Non-K)	0.94	0.95	0.97
average f-measure	0.58	0.60	0.66
weighted f-measure	0.92	0.93	0.96
G-mean	0.83	0.83	0.75

As explained in section IV, however, title words and words in the keyword list have been used as proxy labels. In other words, when we say that a "non-keyword" has been mistakenly classified as a keyword, it is possible that the "non-keyword" did not appear in the title, nor in the keyword list, and yet appeared quite frequently in the entire document, and can be found in the abstract and section headers as well as in some of the figure/table captions. Perhaps that same "non-keyword" has been italicized and printed in large bold-faced font. As far as its features are concerned, it can really look like any of the important words that were correctly classified as "keywords".

Obviously then, there would be many other words in a long document that can be considered important, but these words may not figure among the title words and the keyword list supplied by the authors of the journal paper. This is an inherent limitation of the study. We will not be able to compute the absolute precision and recall of our keyword extraction procedure, nor the relevance of the features that we have used, because we do not have access to an authoritative dataset that has a definitive list of keywords per document.

We have, however, manually inspected the "false positives" among non-keywords which were misclassified to be keywords. Practically all are substantial, non-trivial words and really appeared to be important words in the documents where they appeared. We also tried to rank the extracted keywords per document, using the activation value of the output nodes as basis for ranking. The higher the activation value, the lesser was the chance that a word tagged as keyword is in fact a non-keyword.

The various discussions regarding this manual inspection of the documents are beyond the scope of this paper. We should at least mention, however, that we have developed a software module that allowed for a rapid visual inspection of the "importance" of a word in a given document. This is achieved by color coding of words, so its frequency of appearance, scatter and spread, as well as the various formatting that was used, can be rendered visually and hence, a manual inspection of the tagged words could be facilitated. Refer to Fig. 1. Note that words had been stemmed, and so words with the same root/stem (e.g. query, queries, querying) would be tagged as the same word, and be given the same word color.

The manual inspection of the extracted (non)keywords led to the observation that "false positives" tended to occur more frequently among words that registered relatively lower activation values at the output nodes of the trained network. This observation was the basis for using a validation set in Experiment III, which then would enable us to determine at which threshold value should the output nodes be set, so that the ratio of "true positives" and "false positives" could be maximized, and thus improve the precision for keywords, and therefore maximize the *weighted f-measure*.

By using a validation set during training, the number of false positives among non-keywords is drastically reduced to only 3,207 in Experiment III, compared to 7,193 in Experiment II. This resulted in the improvement of the precision for keywords from 0.16 to 0.24, although, as expected, the accuracy of predicting keywords (*recall*) went significantly lower to 0.58 from 0.76. The precision among non-keywords remained the same, while *recall* among non-keywords improved from 0.91 to 0.96.

Individual *recall* and *precision* rates are helpful in evaluating the accuracy of the trained classifier on a per category basis (i.e. keyword or non-keyword). Since the test set is highly imbalanced, with 98% of the words being non-keywords, it is important to see the *recall* and *precision* separately, as it is done in Table II.

Reporting the performance of any trained classifier is, however, best done using aggregated and summative measures such as the *f measure*, which combines *recall* (r) and *precision* (p) in a single measure, as follows:

$$\text{f measure} = 2rp/(r+p) \quad (5)$$

We see from Table II that the *f-measure* for non-keywords is consistently better than that for keywords, and that the "calibration" using the validation set in Experiment III has actually helped in getting better *f-measures*.

We can aggregate further the *f-measures* of keywords and non-keywords into a single *f-measure*. One way is just to take a simple average (*average f-measure*). The other way is to consider the ratio of keywords to non-keywords and to use a weighted mean. This yields the *weighted f measure* in Table II. Once again, for both consolidated *f-measures*, the "calibrated neural network" of Experiment III produced the best results.

We studied yet another aggregated performance measure, referred to as the *Geometric mean* (**G mean**) and computed as shown in (6). This was first introduced by [37][38], and was specifically focused on reporting the classification performance on highly imbalanced datasets.

$$\text{G mean} = \sqrt{recall(K) \times recall(NonK)} \qquad (6)$$

Note that since the *recall* among keywords dropped for experiment III, the resultant *G mean* has actually decreased to 0.75 compared to 0.83 for experiment II.

As a final step, it is important to figure out the "logic" that the trained networks are using in classifying a given word as keyword or not. This is a field of research in artificial neural networks that is referred to as "rule extraction" [39]. To extract rules from a trained network, we take the initial labels of the word vectors in the test set and replace them with the label (keyword or non-keyword), as predicted by the trained network. This way, the word vectors are now labeled with the neural network predicted label. When we then subject the "re-labeled" test set" to the C4.5 decision tree classifier [40][41], what we get is a decision tree that actually reflects the "rules" that the trained neural network was using. This is a subtle, simple, fast, but effective way of extracting rules from trained neural networks.

By feeding a decision tree classifier with a re-labeled test set, we were able to see which features were deemed important for the backpropagation neural network. The most important features according to the tree are the following: *frequency-in-segment-1*, *frequency-count*, *frequency-in-segment-4*, *usage-in-caption*, *frequency-in-segment-5*, *position-in-sentence*, and *position-in-paragraph*. Figure 2 shows the resulting decision tree from the C4.5 algorithm, giving the details as to how the above features have been used by C4.5 (and the trained network) in arriving at a conclusion, as to whether a word is a keyword or a non-keyword. Due to limitations in space, further discussion on the extracted rules from the resulting decision-tree, particularly in comparison to the rules extracted from neural networks trained using the Wikipedia dataset, would be beyond the scope of this paper. Suffice it to say that the classification performance rates for predicting important from non-important words in IEEE journal articles, as well as the most important features that were extracted, are different from those for Wiki-documents.

VI. Conclusion and Recommendation

Using word feature vectors extracted from IEEE scientific journal articles as training data, several artificial neural networks have been trained using the *backpropagation* learning algorithm. Word features used were limited to only frequency, position, usage and format features – with no attempt to understand the individual words, nor to do any syntactic, much less sematic analysis of the document.

Recall rates for keywords are at around 0.75, while *recall* rates for non-keywords are at around 0.91. The classifiers produced are also highly precise when classifying a word as non-keyword. *Precision* rates are consistently at 0.99. The *precision* for keywords is rather low, however, at around 0.16. Many "non-keywords" have been tagged as important keywords by the classifier. In the study, we used proxy keyword labels in that any word that is in the title or is among the list of keywords supplied just after the abstract of a journal article, is considered a "keyword". All other words in the document are non-keywords. The poor *precision* for the case of keywords can thus be explained partly by the inherent limitation of the study, which had to rely on proxy labels, because no real keyword labels are available.

In an attempt to improve the precision of the neural network in identifying keywords, the threshold for the activation values of the output node of the trained neural network was calibrated during training through the use of a separate validation set. In general, and as expected, as the threshold approaches 1.0, the *precision* increases, but the *recall* rate decreases.

Various aggregated performance measures were also used to assess the classification performance of the neural networks in the three performed experiments. The *recall* and *precision* rates were combined into an *f-measure* for keywords, and an *f-measure* for non-keywords. The *f measures* are further combined into an *average f measure*, and a weighted mean of f-measures (*weighted f-measure*) to take into account the huge imbalance in the dataset, where less than 2% of the words are keywords. Another special performance measure was computed, the *Geometric mean* (*G mean*), which is suitable for imbalanced datasets. A *G mean* of 0.83 was obtained.

Several possibilities for future research include 1) comparison of the "learned" basis for extracting keywords from journal articles with those from websites/homepages and snippets from social media, 2) use of data clustering and self-organizing maps that would render the keywords in a 2D or 3D map, rather than a plain list of extracted keywords; and 3) detecting entire key-phrases, as a sequence of keywords (bi-grams, tri-grams).

References

[1] M.A. Hearst, "Untangling Text Data Mining", Proc Annual Meeting Association for Computational Linguistics, 1999 (invited paper).

[2] I.H. Witten. "Text mining". M. Singh (Ed.), Practical handbook of internet computing (p. 14-1 - 14-22), 2005, Boca Raton, Florida: Chapman & Hall/CRC Press.

[3] R.M. Aliguliyev "Clustering of document collection - A weighting approach". Expert Systems with Applications. V36 I4. 2009, pp 7904-7916.

[4] J. Chuang, C.D. Manning and J. Heer, "Without the clutter of unimportant words: Descriptive keyphrases for text visualization", ACM Trans Computer-Human Interaction , V19 I3, October 2012

[5] S.Y. Bong and K.B. Hwang, "Keyphrase extraction in biomedical publications using mesh and intraphrase word co-occurrence information", Proc ACM 5th Intl Workshop Data and Text Mining in Biomedical Informatics, October 2011.

[6] C-H Wang, M-Z Zhang, L-Y Ru and S-P Ma, "An automatic online news topic keyphrase extraction system, Proc IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2008

[7] Z. Ding, Q. Zhang and X Huang, "Learning to extract coherent keyphrases from online news", Proce 7th Asia Conference on Information Retrieval Technology, 2011, Springer-Verlag

[8] F. Zhang, G. Qiu, J. Bu, M. Qu and C. Chen "A novel approach to keyword extraction for contextual advertising". Proc. Asian Conference on Intelligent Information and Database Systems, 2009, pp 51-56.

[9] S-T Li and F-C Tsai, "Constructing tree-based knowledge structures from text corpus", Applied Intelligence, 33 (1), 2010, p.67-78

[10] L. Massey, "Autonomous and adaptive identification of topics in unstructured text", Proc Intl Conf Knowledge-Based and Intelligent Information and Engineering Systems, 2011

[11] K.i Zervanou, "UvT: The UvT term extraction system in the keyphrase extraction task", SemEval '10: Proc Intl ACL Workshop on Semantic Evaluation, July 2010

[12] D. X. Wang, X. Gao and P. Andreae, "DIKEA: domain-independent keyphrase extraction algorithm", Proc Australasian Joint Conf on Advances in Artificial Intelligence, 2012, Springer-Verlag

[13] S R. El-Beltagy and A Rafea, "KP-Miner: A keyphrase extraction system for English and Arabic documents", Information Systems , 34 (1), 2009, Elsevier Science

[14] S.T. Yanga, J.L. Houb and J.Y. Chenb , "A knowledge component extraction technology using figures and tables", Journal of Experimental & Theoretical Artificial Intelligence, 25(2), 2013, pp 147-175

[15] P. Thomson "A combination of expert opinion approach to probabilistic information retrieval, Part 1: The conceptual model", Information Processing & Management, 26 (3), 1990, pp. 371-382, Pergamon Press

[16] L. Sbattella and R. Tedesco, "A novel semantic information retrieval system based on a three-level domain model", Journal of Systems and Software, 86 , 2013, pp 1426– 1452

[17] A.P. Azcarraga and T.N. Yap Jr., "Extracting meaningful labels for Websom text archives". Proc. Intl Conf on Information and Knowledge Management, 2001, pp 41–48.

[18] M.S. Paukkeri, A.P. García-Plaza, V. Fresno, R.M. Unanue and T. Honkela, "Learning a taxonomy from a set of text documents", Applied Soft Computing , 12 (3) 2012, Elsevier Science

[19] C-H Chou, C-C Han and Y-H Chen, "GA based optimal keyword extraction in an automatic Chinese web document classification system". In ISPA 2007 Workshops, LNCS 4743, (2007) pp. 224–234.

[20] M. Efron," Linear time series models for term weighting in information retrieval", Journal of the American Society for Information Science and Technology, 61 (7),2010, pp.1299-1312, 2010 [doi>10.1002/asi.v61:7]

[21] C. Wu, M. Marchese, Y. Wang, M. Krapivin, C. Wang, X. Li and Y. Liang "Data preprocessing in SVM-based keywords extraction from scientific documents". Proc. Intl Conf on Innovative Computing, Information and Control, 2009, pp 810–813.

[22] W-J Ni, T-L Liu and Q-T Zeng "Extracting keyphrase set with high diversity and coverage using structural SVM", Proc Asia-Pacific Intl Conf on Web Technologies and Applications, 2012, Springer-Verlag.

[23] K. Zhang, H. Xu, J. Tang and J. Li. "Keyword extraction using support vector machine". Proc 7th international conference on Advances in Web-Age Information Management, LNCS 4016, 2006, pp. 85-96. Springer-Verlag.

[24] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators", Neural Networks, 2(5), 1989, pp. 359-366.

[25] A.P. Azcarraga, M.D. Liu and R. Setiono, "Keyword extraction using backpropagation neural networks and rule extraction". Proc International Joint Conference on Neural Networks (IJCNN 2012), Brisbane, Australia.

[26] T.W.S Chow, H. Zhang and M.K.M Rahman, "A new document representation using term frequency and vectorized graph connectionists with application to document retrieval". Expert Systems with Applications. 36 (10), 2009, pp 12023-12035.

[27] T. Jo, M. Lee and T.M. Gatton. "Keyword extraction from documents using a neural network model". Proc. IEEE Intl Conf on Hybrid Information Technology, 2006, pp 194–197

[28] E.D. Wiener, J.O. Pedersen and A.S. Weigend "A neural network approach to topic spotting". Proc. Annual Symposium on Document Analysis and Information Retrieval, 1995, pp 317-322.

[29] T.D. Nguyen and MY Kan "Keyphrase extraction in scientific publications", ICADL'07: Proc 10th international conference on Asian digital libraries, 2007, Springer-Verlag

[30] K. Niraj and S. Kannan, "Automatic keyphrase extraction from scientific documents using N-gram filtration technique", Proc 8th ACM Symposium on Document Engineering

[31] S-N Kim and M-Y Kan, "Re-examining automatic keyphrase extraction approaches in scientific articles", Proc Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications, 2009, Association for Computational Linguistics

[32] A. Joorabchi and A.E. Mahdi, "Automatic keyphrase annotation of scientific documents using Wikipedia and genetic algorithms", Journal of Information Science , 39(3), 2013, Sage

[33] M.F. Porter, "An algorithm for suffix stripping". Readings in Information Retrieval, 1997, pp 313-316. 1997, Morgan Kaufmann

[34] G. Salton, A. Wong and C.S. Yang, "A vector space model for automatic indexing". Communications of the ACM. 18(11), 2005, pp 613-620

[35] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval", Information Processing and Management: an International Journal, 24 (5), 1988, pp.513-523.

[36] I Mierswa, M Wurst, R Klinkenberg, M Scholz and T Euler, "YALE: Rapid Prototyping for Complex Data Mining Tasks", Proc ACM SIGKDD Intl Conf Knowledge Discovery and Data Mining, 2006.

[37] M.K. Robert, R.Holte and S. Matwin, "Learning when negative examples abound". Proc European Conference on Machine Learning, 1997

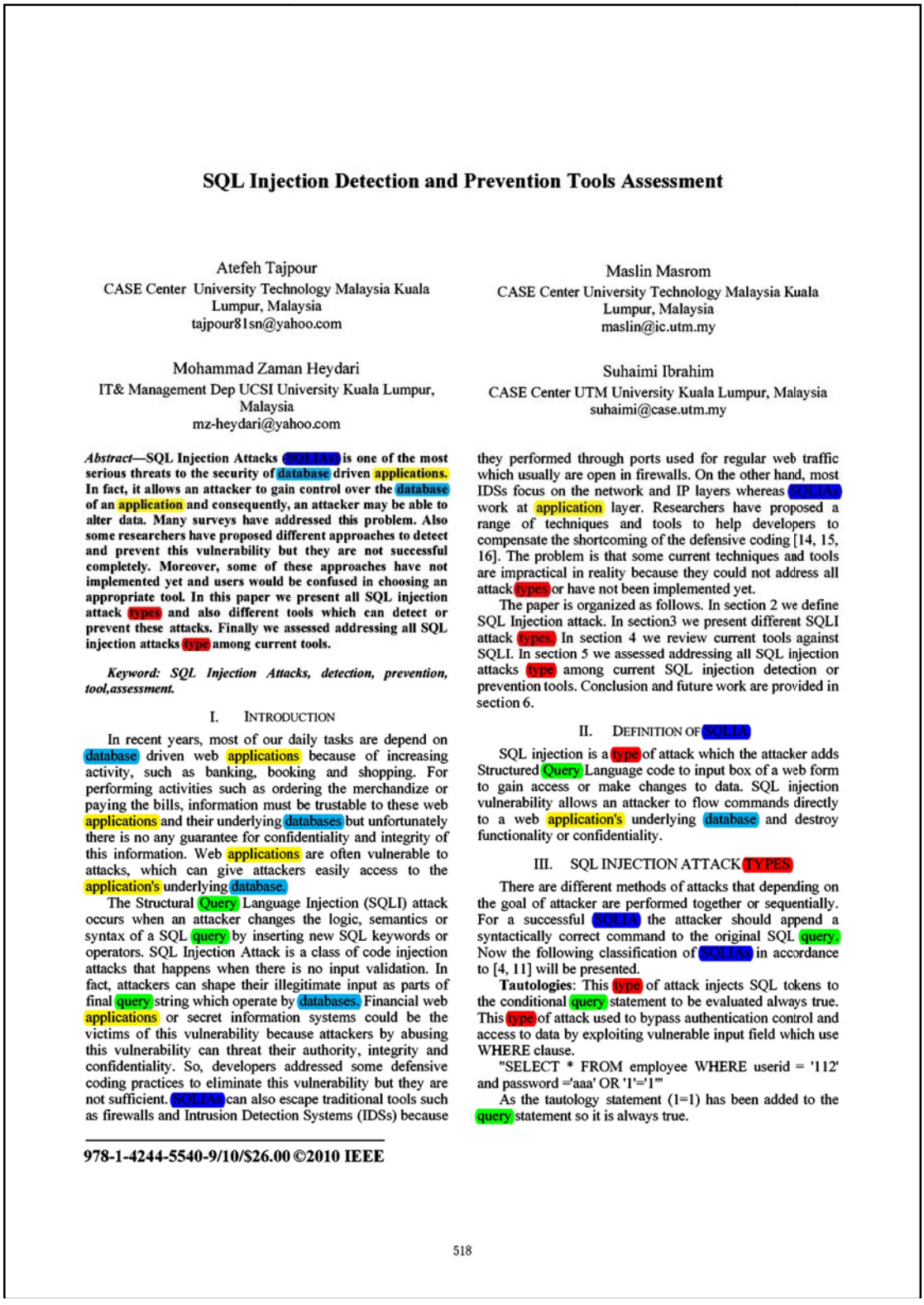
[38] M Kubat and S. Matwin "Addressing the curse of imbalanced training sets: one-sided selection", Proc Intl Conf on Machine Learning, 1997, Morgan Kaufmann

[39] R. Setiono, B. Baesens and C. Mues, "Recursive neural network rule extraction for data with mixed attributes", IEEE Trans. on Neural Networks, 19(2), 2008, pp 299–307.

[40] J.R. Quinlan, "C4.5: Programs for machine learning". 1993, Morgan Kaufmann

[41] J.R. Quinlan "Improved use of continuous attributes in C4.5". Journal of Artificial Intelligence Research, 4, 1996, pp. 77-90.

[42] H. Goto, Y. Hasegawa and M. Tanaka, "Efficient Scheduling Focusing on the Duality of MPL Representatives," Proc. IEEE Symp. Computational Intelligence in Scheduling (SCIS 07), IEEE Press, Dec. 2007, pp. 57-64, doi:10.1109/SCIS.2007.357670.



SQL Injection Detection and Prevention Tools Assessment

Atefeh Tajpour
CASE Center University Technology Malaysia Kuala Lumpur, Malaysia
tajpour81sn@yahoo.com

Maslin Masrom
CASE Center University Technology Malaysia Kuala Lumpur, Malaysia
maslin@ic.utm.my

Mohammad Zaman Heydari
IT& Management Dep UCSI University Kuala Lumpur, Malaysia
mz-heydari@yahoo.com

Suhaimi Ibrahim
CASE Center UTM University Kuala Lumpur, Malaysia
suhaimi@case.utm.my

***Abstract*—SQL Injection Attacks (SQLIAs) is one of the most serious threats to the security of database driven applications. In fact, it allows an attacker to gain control over the database of an application and consequently, an attacker may be able to alter data. Many surveys have addressed this problem. Also some researchers have proposed different approaches to detect and prevent this vulnerability but they are not successful completely. Moreover, some of these approaches have not implemented yet and users would be confused in choosing an appropriate tool. In this paper we present all SQL injection attack types and also different tools which can detect or prevent these attacks. Finally we assessed addressing all SQL injection attacks type among current tools.**

Keyword: SQL Injection Attacks, detection, prevention, tool,assessment.

I. INTRODUCTION

In recent years, most of our daily tasks are depend on database driven web applications because of increasing activity, such as banking, booking and shopping. For performing activities such as ordering the merchandize or paying the bills, information must be trustable to these web applications and their underlying databases but unfortunately there is no any guarantee for confidentiality and integrity of this information. Web applications are often vulnerable to attacks, which can give attackers easily access to the application's underlying database.

The Structural Query Language Injection (SQLI) attack occurs when an attacker changes the logic, semantics or syntax of a SQL query by inserting new SQL keywords or operators. SQL Injection Attack is a class of code injection attacks that happens when there is no input validation. In fact, attackers can shape their illegitimate input as parts of final query string which operate by databases. Financial web applications or secret information systems could be the victims of this vulnerability because attackers by abusing this vulnerability can threat their authority, integrity and confidentiality. So, developers addressed some defensive coding practices to eliminate this vulnerability but they are not sufficient. SQLIAs can also escape traditional tools such as firewalls and Intrusion Detection Systems (IDSs) because they performed through ports used for regular web traffic which usually are open in firewalls. On the other hand, most IDSs focus on the network and IP layers whereas SQLIAs work at application layer. Researchers have proposed a range of techniques and tools to help developers to compensate the shortcoming of the defensive coding [14, 15, 16]. The problem is that some current techniques and tools are impractical in reality because they could not address all attack types or have not been implemented yet.

The paper is organized as follows. In section 2 we define SQL Injection attack. In section3 we present different SQLI attack types. In section 4 we review current tools against SQLI. In section 5 we assessed addressing all SQL injection attacks type among current SQL injection detection or prevention tools. Conclusion and future work are provided in section 6.

II. DEFINITION OF SQLIA

SQL injection is a type of attack which the attacker adds Structured Query Language code to input box of a web form to gain access or make changes to data. SQL injection vulnerability allows an attacker to flow commands directly to a web application's underlying database and destroy functionality or confidentiality.

III. SQL INJECTION ATTACK TYPES

There are different methods of attacks that depending on the goal of attacker are performed together or sequentially. For a successful SQLIA the attacker should append a syntactically correct command to the original SQL query. Now the following classification of SQLIAs in accordance to [4, 11] will be presented.

Tautologies: This type of attack injects SQL tokens to the conditional query statement to be evaluated always true. This type of attack used to bypass authentication control and access to data by exploiting vulnerable input field which use WHERE clause.

"SELECT * FROM employee WHERE userid = '112' and password ='aaa' OR '1'='1'"

As the tautology statement (1=1) has been added to the query statement so it is always true.

978-1-4244-5540-9/10/$26.00 ©2010 IEEE

Figure 1. Visualization tool that assigns color-codes to tagged keywords, so their frequency of appearance, scatter and spread, as well as the various formatting that was used, can be readily seen and observed, thus facilitating the manual inspection of the tagged words. Note that words had been stemmed, and so words with the same root/stem (e.g. query, queries, querying) would be tagged as the same word and are assigned the same word color.

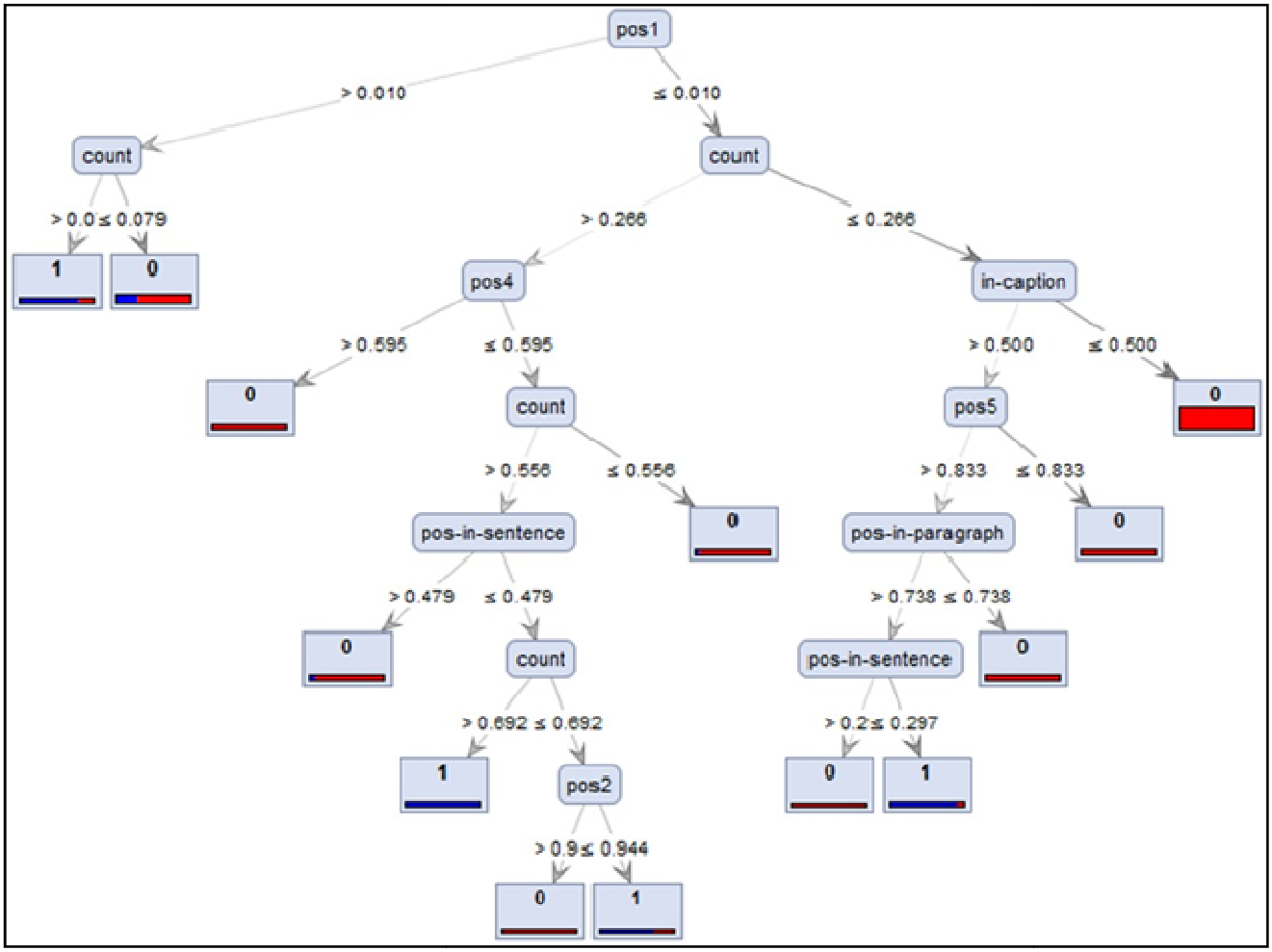


Figure 2. Decision tree generated with C4.5 based on the neural network classifications. The resultant tree reflects the "rules" used by the neural network in classifying a given word as either a keyword or non-keyword. Class 0 refers to non-keywords, class 1 refers to keywords.