ISSN 1870-4069

Extractive Summarization using Deep Learning

Sukriti Verma, Vagisha Nidhi

Delhi Technological University,
India
sukriti_bt2k14@dtu.ac.in,vagisha.nda@gmail.com

Abstract. This paper proposes a text summarization approach for factual reports using a deep learning model. This approach consists of three phases: feature extraction, feature enhancement, and summary generation, which work together to assimilate core information and generate a coherent, understandable summary. We are exploring various features to improve the set of sentences selected for the summary, and are using a Restricted Boltzmann Machine to enhance and abstract those features to improve resultant accuracy without losing any important information. The sentences are scored based on those enhanced features and an extractive summary is constructed. Experimentation carried out on several articles demonstrates the effectiveness of the proposed approach.

Keywords: unsupervised, single document, deep learning, extractive.

1 Introduction

A summary can be defined as a text produced from one or more texts, containing a significant portion of the information from the original text(s), and that is no longer than half of the original text(s) [1]. According to [2], text summarization is the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user and task(s). When this is done by means of a computer, i.e. automatically, we call it Automatic Text Summarization. This process can be seen as a form of compression and it necessarily suffers from information loss but it is essential to tackle the information overload due to abundance of textual material available on the Internet.

Text Summarization can be classified into extractive summarization and abstractive summarization based on the summary generated. Extractive summarization is creating a summary based on strictly what you get in the original text. Abstractive summarization mimics the process of paraphrasing a text. Text(s) summarized using this technique looks more human-like and produces condensed summaries. These techniques are much harder to implement than the extractive summarization techniques.

In this paper, we follow the extractive methodology to develop techniques for summarization of factual reports or descriptions. We have developed an approach for single-document summarization using deep learning. So this paper intends to propose an approach by referencing the architecture of the human brain. It is broken down into three phases: feature extraction [3], feature enhancement, and summary generation based on values of those features. Since it can be very difficult to construct high-level, abstract features from raw data, we use deep learning in the second phase to build complex features out of simpler features extracted in the first phase. These extracted features depend highly on how factual the given document is. In the end, we have run the proposed algorithm on several factual reports to evaluate and demonstrate the effectiveness of the proposed approach based on the measures such as Recall, Precision, and F-measure.

2 Related Works

Most early work on text summarization was focused on technical documents and early studies on summarization aimed at summarizing from pre-given documents without any other requirements, which is usually known as generic summarization [4]. Luhn [5] proposed that the frequency of a particular word in an article provides a useful measure of its significance. A number of key ideas, such as stemming and stop word filtering, were put forward in this paper that have now been understood as universal preprocessing steps to text analysis. Baxendale [6] examined 200 paragraphs and found that in 85% of the paragraphs, the topic sentence came as the first one and in 7% of the time, it was the last sentence. This positional feature has been used in many complex machine learning based systems since. Edmundson [7] focused his work around the importance of word frequency and positional importance as features. Two other features were also used: cue words, and the skeleton structure of the document. Weights were associated with these features manually and finally sentences were scored. During evaluation, it was found that around 44% of the system generated summaries matched the target summaries written manually by humans.

Upcoming researchers in text summarization have approached it problem from many aspects such as natural language processing [8], statistical modelling [9] and machine learning. While initially most machine learning systems assumed feature independence and relied on naive-Bayes methods, other recent ones have shifted focus to selection of appropriate features and learning algorithms that make no independence assumptions. Other significant approaches involved Hidden Markov Models and log-linear models to improve extractive summarization. More recent papers, in contrast, used neural networks towards this goal.

Text Summarization can be done for one document, known as single-document summarization [10], or for multiple documents, known as multi-document summarization [11]. On basis of the writing style of the final summary generated, text summarization techniques can be divided into extractive methodology and abstractive methodology [12]. The objective of generating summaries via the extractive approach is choosing certain appropriate sentences as per the requirement of a user. Due to the idiosyncrasies of human-invented languages and grammar, extractive approaches, which select a subset of sentences from the input documents to form a summary instead of paraphrasing like a human [13], are the mainstream in the area.

Almost all extractive summarization methods have three main obstacles. The first obstacle is the ranking problem i.e. how you rank words, phrases and/or sentences. The second obstacle is the selection problem i.e. how to select a subset of those ranked units [14]. The third obstacle is the coherence problem i.e. how to ensure that the selected units form an understandable summary rather than being a set of disconnected words, phrases and/or sentences. Algorithms that determine the relevance of a textual unit, that is words, phrases and/or sentences, with respect to the requirement of the user are used to solve the ranking problem. The selection and coherence problems are solved by methods that improve diversity, minimize redundancy and pick up phrases and/or sentences that are somewhat similar so that more relevant information can be covered by the summary in lesser words and the summary is coherent. Our approach solves the ranking problem by learning a certain set of features for each sentence. On the basis of these features, a score is calculated for each sentence and sentences are arranged in decreasing order of their scores [15]. Even with a list of ranked sentences, it is not a trivial problem to select a subset of sentences for a coherent summary which includes diverse information, minimizes redundancy and is within a word limit. Our approach solves this problem as follows. The most relevant sentence is the first sentence in this sorted list and is chosen as part of the subset of sentences which will form the summary. Then the next sentence selected is a sentence having highest Jaccard similarity with the first sentence and is picked from the top half of the list. This process is recursively and incrementally repeated to select more sentences until limit is reached.

3 Proposed Approach

3.1 Preprocessing

Preprocessing is crucial when it comes to processing text. Ambiguities can be caused by various verb forms of a single word, different accepted spellings of a certain word, plural and singular terms of the same things. Moreover, words like a, an, the, is, of etc. are known as stop words. These are certain high frequency words that do not carry any information and don't serve any purpose towards our goal of summarization. In this phase we do:

1. **Document Segmentation:** The text is divided into paragraphs so as to keep a track of which paragraph each sentence belongs to and what is the position of a sentence in its respective paragraph.
2. **Paragraph Segmentation:** The paragraphs are further divided into sentences.
3. **Word Normalization:** Each sentence is broken down into words and the words are normalized. Normalization involves lemmatization and results in all words being in one common verb form, crudely stemmed down to their roots with all ambiguities removed. For this purpose, we use Porters algorithm.

4. **Stop Word Filtering:** Each token is analyzed to remove high frequency stop words.
5. **PoS Tagging:** Remaining tokens are Part-of-Speech tagged into verb, noun, adjective etc. using the PoS Tagging module supplied by NLTK [16].

3.2 Feature Extraction

Once the complexity has been reduced and ambiguities have been removed, the document is structured into a sentence-feature matrix. A feature vector is extracted for each sentence. These feature vectors make up the matrix. We have experimented with various features. The combination of the following 9 sentence features has turned out most suitable to summarize factual reports. These computations are done on the text obtained after the preprocessing phase:

1. **Number of thematic words:** The 10 most frequently occurring words of the text are found. These are thematic words. For each sentence, the ratio of no. of thematic words to total words is calculated.

$$Sentence\_Thematic = \frac{No.\ of\ thematic\ words}{Total\ words}. \tag{1}$$

2. **Sentence position:** This feature is calculated as follows.

$$Sentence\_Position = \begin{cases} 1, \text{if its the first or last sentence of the text,} \\ cos((SenPos - min)((1/max) - min)), \text{otherwise,} \end{cases} \tag{2}$$

where, SenPos = position of sentence in the text
min = th x N
max = th x 2 x N

N is total number of sentences in document
th is threshold calculated as 0.2 x N
By this, we get a high feature value towards the beginning and ending of the document, and a progressively decremented value towards the middle.

3. **Sentence length:** This feature is used to exclude sentences that are too short as those sentences will not be able to convey much information.

$$Sentence\_Length = \begin{cases} 0, \text{if number of words is less than 3,} \\ No.\ of\ words\ in\ the\ sentence, \text{otherwise.} \end{cases} \tag{3}$$

4. **Sentence position relative to paragraph:** This comes directly from the observation that at the start of each paragraph, a new discussion is begun and at the end of each paragraph, we have a conclusive closing.

$$Position\_In\_Para = \begin{cases} 1, \text{if it is the first or last sentence of a paragraph,} \\ 0, \text{otherwise.} \end{cases} \tag{4}$$

5. **Number of proper nouns:** This feature is used to give importance to sentences having a substantial number of proper nouns. Here, we count the total number of words that have been PoS tagged as proper nouns for each sentence.
6. **Number of numerals:** Since figures are always crucial to presenting facts, this feature gives importance to sentences having certain figures. For each sentence we calculate the ratio of numerals to total number of words in the sentence.

$$Sentence\_Numerals = \frac{No.\ of\ numerals}{Total\ words}. \tag{5}$$

7. **Number of named entities:** Here, we count the total number of named entities in each sentence. Sentences having references to named entities like a company, a group of people etc. are often quite important to make any sense of a factual report.
8. **Term Frequency-Inverse Sentence Frequency (TF − ISF):** Since we are working with a single document, we have taken TF-ISF feature into account rather than TF-IDF. Frequency of each word in a particular sentence is multiplied by the total number of occurrences of that word in all the other sentences. We calculate this product and add it over all words.

$$TF - ISF = \frac{log(\sum_{all\ words} TF * ISF)}{Total\ words}. \tag{6}$$

9. **Sentence to Centroid similarity:** Sentence having the highest TF-ISF score is considered as the centroid sentence. Then, we calculate cosine similarity of each sentence with that centroid sentence.

$$Sentence\_Similarity = cosine\_sim(sentence, centroid). \tag{7}$$

At the end of this phase, we have a sentence-feature matrix.

3.3 Feature Enhancement

The sentence-feature matrix has been generated with each sentence having 9 feature vector values. After this, recalculation is done on this matrix to enhance and abstract the feature vectors, so as to build complex features out of simple ones. This step improves the quality of the summary.

To enhance and abstract, the sentence-feature matrix is given as input to a Restricted Boltzmann Machine (RBM) which has one hidden layer and one visible layer. A single hidden layers will suffice for the learning process based on the size of our training data. The RBM that we are using has 9 perceptrons in each layer with a learning rate of 0.1. We use Persistent Contrastive Divergence method to sample during the learning process [17]. We have trained the RBM for 5 epochs with a batch size of 4 and 4 parallel Gibbs Chains, used for sampling using Persistent CD method. Each sentence feature vector is passed through the

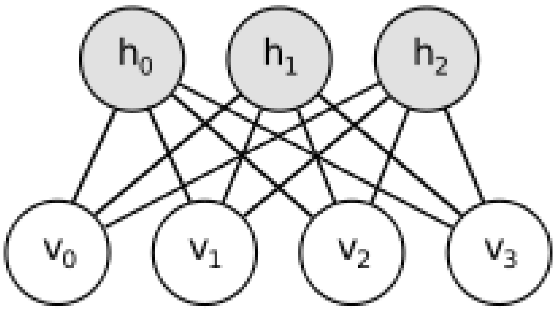

Fig. 1. A Restricted Boltzmann Machine [17].

hidden layer in which feature vector values for each sentence are multiplied by learned weights and a bias value is added to all the feature vector values which is also learned by the RBM. At the end, we have a refined and enhanced matrix. Note that the RBM will have to be trained for each new document that has to be summarized. The idea is that no document can be summarized without going over it. Since each document is unique in the features extracted in section 3.2, the RBM will have to be freshly trained for each new document.

3.4 Summary Generation

The enhanced feature vector values are summed to generate a score against each sentence. The sentences are then sorted according to decreasing score value. The most relevant sentence is the first sentence in this sorted list and is chosen as part of the subset of sentences which will form the summary. Then the next sentence we select is the sentence having highest Jaccard similarity with the first sentence, selected strictly from the top half of the sorted list. This process is recursively and incrementally repeated to select more sentences until a user-specified summary limit is reached. The sentences are then re-arranged in the order of appearance in the original text. This produces a coherent summary rather than a set of haywire sentences.

4 Results and Performance Evaluation

Several factual reports from various domains of health, technology, news, sports etc. with varying number of sentences were used for experimentation and evaluation. The proposed algorithm was run on each of those and system-generated summaries were compared to the summaries produced by humans.

Feature Extraction and Enhancement is carried out as proposed in sections 3.2 and 3.3 for all documents. The values of feature vector sum and enhanced feature vector sum for each sentence of one such document have been plotted in Fig 2. The Restricted Boltzmann Machine has extracted a hierarchical representation out of data that initially did not have much variation, hence discovering

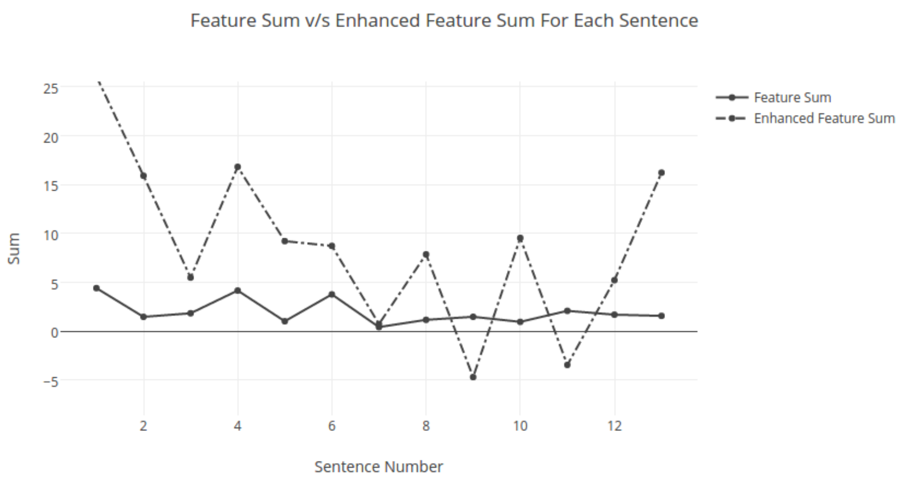

Fig. 2. Comparison between feature vector sum and enhanced feature vector sum.

the latent factors. The sentences have then been ranked on the basis of final feature vector sum and summaries are generated as proposed in section 3.4.

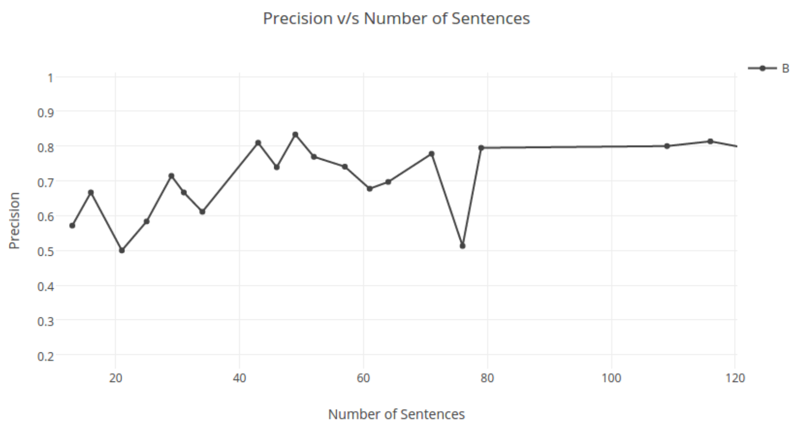

Fig. 3. Precision values corresponding to summaries of various documents.

Evaluation of the system-generated summaries is done based on three basic measures: Precision, Recall and F-Measure [18].

It can be seen that as the number of sentences in the original document cross a certain threshold, the Restricted Boltzmann Machine has ample data to

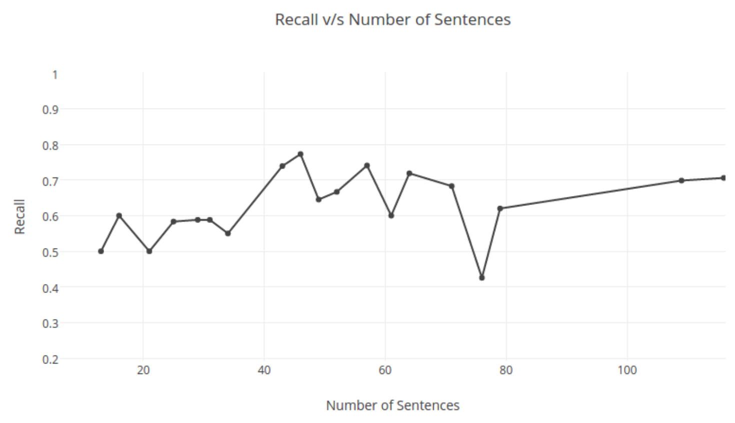

Fig. 4. Recall values corresponding to summaries of various documents.

be trained successfully and summaries with high precision and recall values are generated. See Fig 3 and 4.

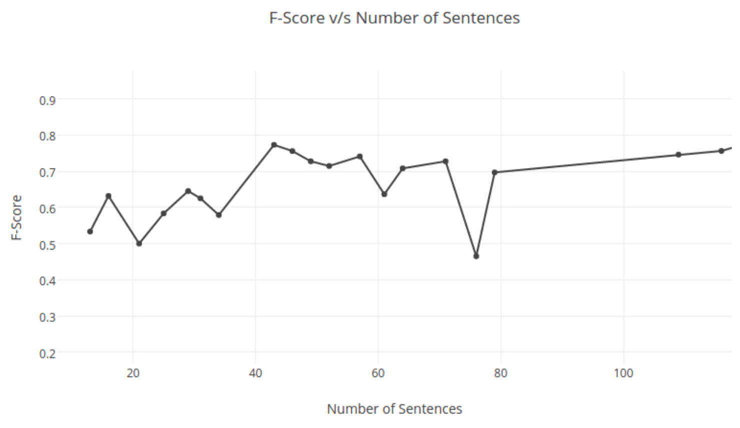

Fig. 5. F-Measure values corresponding to summaries of various documents.

F-Measure is defined as follows [19]:

$$F - Measure = \frac{2 * Recall * Precision}{Recall + Precision} \tag{8}$$

5 Comparative Analysis

The existing approach was executed for the same set of articles with just one layer of RBM, rather than two as it specifies and average values of Precision, Recall and F-Measure were plotted for drawing a comparison between the existing approach and the proposed approach, while keeping the amount of computation constant.

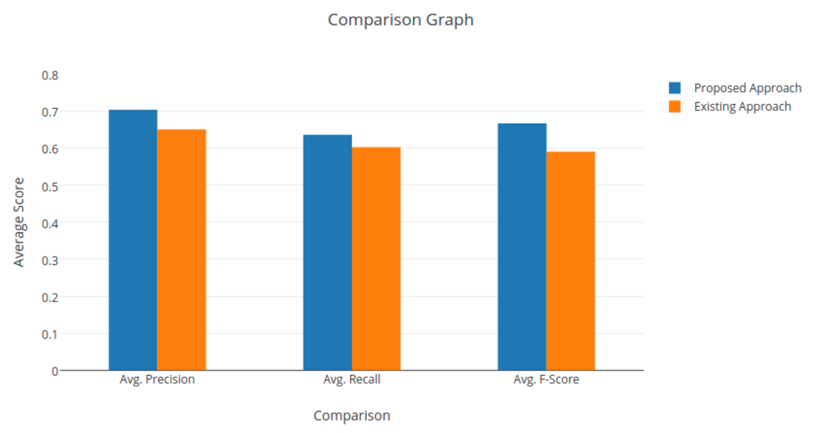

Fig. 6. Precison, Recall and F-Measure values for the proposed approach (*left bars*) and the existing approach (*right bars*).

The proposed approach has an average precision value of 0.7 and average recall value of 0.63 which are both higher than those of the existing approach. Hence, the proposed approach responds better for summarization of factual reports.

6 Conclusion

We have developed an algorithm to summarize single-document factual reports. The algorithm runs separately for each input document, instead of learning rules from a corpus, as each document is unique in itself. This is an advantage that our approach provides. We extract 9 features from the given document and enhance them to score each sentence. Recent approaches have been using 2 RBMs stacked on top of each other for feature enhancement. Our approach uses only one RBM and, works effectively and efficiently for factual reports. This has been demonstrated by hand-picking factual descriptions from several domains and comparing the system-generated summaries to those written by humans. This approach can further be developed by adapting the extracted features as per the user's requirements and further adjusting the hyperparameters of the RBM to minimize processing and error in encoded values.

Acknowledgments. We would like to extend our gratitude to Dr. Daya Gupta, Professor, Department of Computer Science and Engineering, Delhi Technological University(Formerly Delhi College of Engineering) for providing insight and expertise that greatly assisted this research.

References

1. Hovy, E. H.: Automated Text Summarization, In Ruslan Mitkov (ed): The Oxford Handbook of Computational Linguistics 583–598 (2005)
2. Mani, I., House, D., Klein, G., Hirschman, L., Firmin, T., Sundheim, B.: The TIPSTER SUMMAC Text Summarization Evaluation. In: Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics, pp. 77–85. Association for Computational Linguistics Stroudsburg, PA, USA (1999). doi: 10.3115/977035.977047
3. Chuang, W.T., Yang, J.: Extracting Sentence Segments for Text Summarization: A Machine Learning approach. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 152–159. ACM New York, NY, USA (2000). doi: 10.1145/345508.345566
4. Berger, A., Mittal ,V.: Query relevant summarization using FAQs. In: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, pp. 294–301. Association for Computational Linguistics Stroudsburg, PA, USA (2000). doi: 10.3115/1075218.1075256
5. Luhn, H. P.: The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development, vol. 2, issue 2, 159–165 (1958). doi: 10.1147/rd.22.0159
6. Baxendale, P.: Machine-Made Index for Technical Literature - An Experiment. IBM Journal of Research and Development, vol. 2, issue 4, 354–361 (1958). doi: 10.1147/rd.24.0354
7. Edmundson, H. P.: New methods in Automatic Extracting. Journal of the ACM, vol. 16, issue 2, 264–285 (1969). doi: 10.1145/321510.321519
8. Zhang, Y., Wang, D., Li, T.: iDVS - An Interactive Multi-Document Visual Summarization System: Machine Learning and Knowledge Discovery in Databases, LNCS, vol. 6913, pp. 569–584. Springer, Heidelberg (2006). doi: 10.1007/978-3-642-23808-6_37
9. Darling, W.M., Song, F.: Probabilistic Document Modeling for Syntax Removal in Text Summarization. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 642–647. ACM Press, Stroudsburg, PA, USA (2011).
10. Wan, X., Xiao, J.: Single Document Keyphrase Extraction using Neighborhood Knowledge. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (2008).
11. Shen, D., Sun, J.T., Li, H., Yang, Q., Chen, Z.: Document Summarization using Conditional Random Fields. In: Proceedings of the 20th international joint conference on Artifical Intelligence, pp. 2862–2867. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA (2007).
12. Wong, K.F., Wu, M.J., Li, W.J.: Extractive Summarization Using Supervised and Semi-supervised Learning. In: Proceedings of the 22nd International Conference on Computational Linguistics – Volume 1, pp. 985–992. Association for Computational Linguistics Stroudsburg, PA, USA (2008).

13. Chen, E.K., Yang, X.K., Zha, H.Y., Zhang, R., and Zhang, W.J. (2008). Learning Object Classes from Image Thumbnails through Deep Neural Networks. In IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE (2008). doi: 10.1109/ICASSP.2008.4517738
14. Jin, F., Huang, M.L., and Zhu, X.Y.: A Comparative Study on Ranking and Selection Strategies for Multi Document Summarization. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters, pp. 525–533. Association for Computational Linguistics Stroudsburg, PA, USA (2010).
15. Singh, S.P., Kumar, A., Mangal, A., Singhal, S.: Bilingual Automatic Text Summarization Using Unsupervised Deep Learning. In: 2016 International Conference on Electrical, Electronics, and Optimization Techniques. IEEE (2016). doi: 10.1109/ICEEOT.2016.7754874
16. Natural Language Toolkit for Python, `http://www.nltk.org/`
17. Deep Learning Tutorials, `http://deeplearning.net/tutorial/`
18. Performance Evaluation of Information Retrieval Systems, `web.stanford.edu/class/cs276/handouts/lecture8-evaluation.ppt`
19. PadmaPriya, G., Duraiswamy, K.: An Approach for Text Summarization using Deep Learning Algorithm. Journal of Computer Science, vol. 10, issue 1, 1–9 (2014). doi: 10.3844/jcssp.2014.1.9