Detection of text-based advertising and promotion in Wikipedia by deep learning method

Yuanzhen Guo, Mizuho Iwaihara

Graduate School of Information, Production and Systems, Waseda University

2-7 Hibikono, Wakamatu-ku, Kitakyushu-shi, Fukuoka-ken, 808-0135 Japan

E-mail: steve_guo@akane.waseda.jp, iwaihara@waseda.jp

Abstract

Wikipedia is an open Internet encyclopedia that everyone can access and edit. Due to its "written from a neutral point of view" policy, both advertising and promotion are strictly forbidden in Wikipedia, and advertising articles will be deleted by administrators manually. Currently most researches about spamming in Wikipedia are focusing on editing behavior and making use of user's edit history to do feature-based judging. In this paper, we propose a pure text-based method to automatically detect advertising and promotion articles in Wikipedia. In order to apply learning algorithms to training corpus, we need to transform text article into a vector form. Rather than traditional bag-of-words document vector representation which was proved to be inefficient, we employ a deep learning method to obtain a word vector for each word and then apply a sliding window on each document to gradually gain the document vector. Furthermore, we implement an improved deep-learning structure which can directly give us a document vector for each document. We then construct a supervised SVM classifier on the document vectors to obtain the final results. Our method was tested on several datasets and produced better performance than the bag-of-words model in both precision and recall measurements.

Keywords: Wikipedia, deep learning, advertisement detection

1 Introduction

Wikipedia is an Internet encyclopedia aiming at facilitating collaboration and information sharing. Wikipedia allows anyone from different backgrounds (even anonymous users) to create and edit articles. Launched in January 2001, Wikipedia has grown into one of the largest free public knowledge bases with about 34 million articles in more than 200 languages. Wikipedia adopts the concept of maintaining a neutral point of view (NPOV) as one of its founding principles. However, Wikipedia's unrestricted editing access to everyone makes this NPOV goal impossible. Instead of collaborating on editing neutral articles, some users regard Wikipedia as a way to advertise or promote their products. Such advertising and promotion is strictly forbidden in Wikipedia. Currently all the deletions of these advertising articles are done by Wikipedia administrators manually. Administrators have to browse the suspected articles themselves and decide whether they are advertising or promotion by their own experiences. This may requires a lot of time and hard work regarding the relatively small number of administrators comparing to the large amount of advertising and promotion articles in Wikipedia. Let alone administrators may make wrong decisions deleting those articles which are actually just states of fact which leads to even worse situation. So Wikipedia's current advertising and promotion article deletion procedure is time consuming and inefficient. An automatic and more efficient way of deleting advertising articles in Wikipedia is needed.

In order to solve this problem, we propose a learning method to automatically detect advertising and promotion articles in Wikipedia. Most of the former researches on Wikipedia spam or vandalism detection are feature-based learning approaches which analyze user's behavior to do judgment. For example, Potthast et al. [10] used machine learning in combination with manually crafted rules to classify Wikipedia spam edits. Itakura et al. [4] employed a compression model-based algorithm to detect spam editing behaviors in Wikipedia. Lam et al. [5] proposed a machine learning approach to implement spam detection by extracting seven features from each email and using supervised learning to learn the behaviors of spammers. However, our method is pure text-based which means that neither edit history nor user behavior is needed. In this way, we can avoid the ad-hoc problem indicating that we do not need to create rules to extract particular features from the data set which may only works well in special environment. Instead our method not only can be applied in Wikipedia study but also can be extended to other big social networks analysis as long as large text corpus is available.

We first implemented an unsupervised deep learning method to achieve fixed-length vector representation for each unique word in the corpus. We use the same deep learning structure as described in Mikolov et al.'s paper [8, 9]. We then calculate the document vector by taking either the mean of word vectors or tf-idf weighted average. Then we apply a supervised SVM classifier to train the model and use the model to detect advertising and promotion articles in the test set. We also try directly getting document vector from texts by modifying the deep learning structure to a new form as introduced by Mikolov [6]. Finally we improve the method by using LDA words in topic distributions as the initial word vectors. Our method showed better performance on several datasets than bag-of-words models in both precision and recall measurements.

The rest of this paper is organized as follows: In Section 2 we survey related works, including vector space model, Word to Vector model and Document to Vector model. In Section 3 we discuss how our advertising detection system works in details, including pre-processing of the corpus, learning the document vector for each document using Word2Vec, training different classifiers and detection of advertising articles on the test set. Section 4 shows experiments and results, which compares two methods: Bag of Words and Word to Vector, and analyze performance of these two methods. Section 5 is the conclusion and future work.

2 Related work

2.1 Vector Space Model

Vector space model or term vector model is an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers, such as, for example, index terms. It is widely used in natural language processing, information retrieval, indexing and relevancy rankings. Traditionally a word is represented by a one-hot vector where the vector size equals the vocabulary size and the position represents word index is 1 while the others are 0s. However, the one-hot word vector model suffers two main problems: One is that as the size of the data grows the vocabulary size becomes so large, yielding to the curse of dimensionality (Bengio et al. 2003[1]) and the one is that this one-hot representation captures no syntactic or semantic regularities of words since the distances between any two words are the same in the vector space. Then came out the distributed representation of words [1] and it has achieved significant success in the recent past. Instead of a one-hot vector representation, a word is represented by a fixed length (usually several hundreds) real-valued vector. The distributional hypothesis states that words in similar contexts have similar meanings [8]. Intuitively, it means that words who share many same contexts will be similar to each other in the vector space. The distributed representation does not face the-curse-of-dimensionality problem because the length of the vector size is not proportional compared to the data set growth.

2.2 Word2Vec

Word2Vec is an open source project released by Google which achieved state of the art performances in many natural language processing tasks. It takes a large text corpus as input and outputs the word vectors for each unique word. These word vectors can be subsequently used in many natural language processing and machine learning applications such as classification, clustering and other further research. In Mikolov et al.'s word2vec paper [8], they carried out two neural network models for representation learning: Continuous Bag-of-Words Model (CBOW) and Continuous Skip-gram Model. Figure 1 shows the structure of these two models. CBOW uses the word vectors of adjacent words in the range of a surrounding window (e.g. 3 previous words and 3 latter words) to predict the word vector of the central word, while on the contrary Skip-gram uses the central word's vector to predict the surrounding words. As for neural network's optimization and back propagation part, Word2Vec adopts hierarchical softmax based on CBOW and the negative sampling method based on Skip-gram [3]. Hierarchical Softmax was first introduced by Morin and Bengio [1] in the neural network language models. It uses a binary tree (a Huffman tree in Word2Vec) to represent neural network's output layer where words are placed as its leaves and nodes represent relative probabilities. The main advantage of this method is that we only need $\log_2(W)$ nodes along a path to obtain probability distribution instead of W nodes in the standard neural network models. Negative Sampling is a simplified version of Noise Contrastive Estimation (NCE) method [2] which reduces the language model estimation problem to the problem of estimating the parameters of a probabilistic binary classifier and differentiates data from noise. The main difference is that NCE needs both samples and the numerical probabilities of the noise distribution, while Negative Sampling uses only samples.

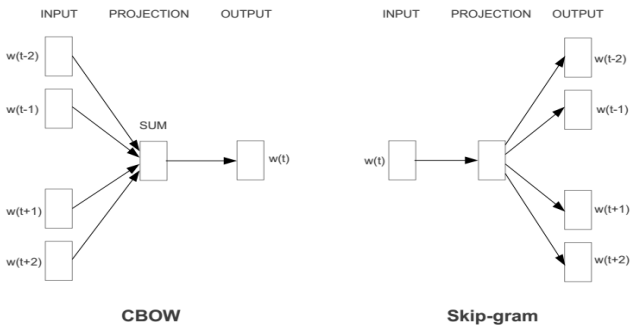

Figure 1 CBOW and Skip-gram model structure

By applying the CBOW and Skip-gram neural network structure, we can achieve very promising results. Similar words will be closer to each other in the vector space after training [11]. For example, if we search for the most similar word to 'France', by just calculating the distances between words in the vector space, we would probably obtain 'Spain', 'Belgium', 'Netherlands', and so on. It was also shown that the word vectors captured many linguistic regularities, for example vector operations: vector ('Paris') – vector ('France') + vector ('Italy') results in a vector that is very close to vector ('Rome'), and vector ('king') – vector ('man') + vector ('woman') is close to vector ('queen').

2.3 Doc2Vec

Doc2Vec (also known as paragraph2vec or sentence embedding) is a modified version of Word2Vec algorithm which carries out unsupervised learning to obtain continuous representations for larger blocks of texts, such as sentences, paragraphs or entire documents. It was first described in Mikolov's latter paper [6]. Figure 2 shows the structure improvement from Word2Vec to Doc2Vec. More formally, the only change in this model compared to the word vector framework is adding all the document vectors of same length to the word vectors into the binary tree as additional

leaves so that we can obtain the document vectors optimized just as the word vectors. In the word2vec architecture, the two algorithm names were Continuous Bag of Words and Skip-gram. In the doc2vec architecture, the corresponding algorithms are Distributed memory and Distributed Bag of Words. A sliding window goes through the document to sample fixed-length contexts for each learning process. The document vector is shared if the texts sampled were sampled from same document but not shared across different documents. However, the word vector is shared across all the documents meaning that there is only one word vector representation for each unique word in the whole corpus.

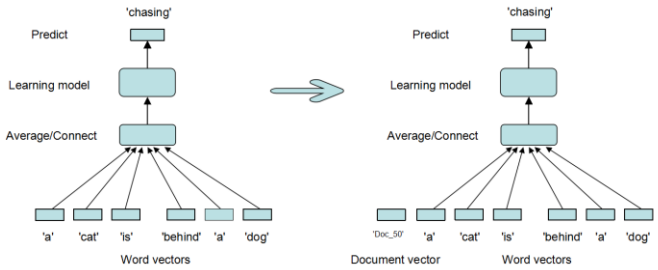

Figure 2 Word2Vec to Doc2Vec

3 Implement of the detection method

3.1 Word Vectors

To begin with, we pre-process the corpus, where all the stop words in articles like 'the', 'a', 'in' etc. are removed, and words with total counts less than 3 times in the corpus are also deleted because these words appear only a few times that are negligible to the whole corpus.

Then Word2Vec algorithm is used to obtain the word vector for each unique word in the corpus. We implement the CBOW plus Hierarchical Softmax model described in the Word2Vec paper [8] which uses 2*N* surrounding words (*N* ahead and *N* behind) to predict the middle word and builds up a Huffman tree to represent probability distribution of words in the corpus. The model structure is shown in Figure 3.

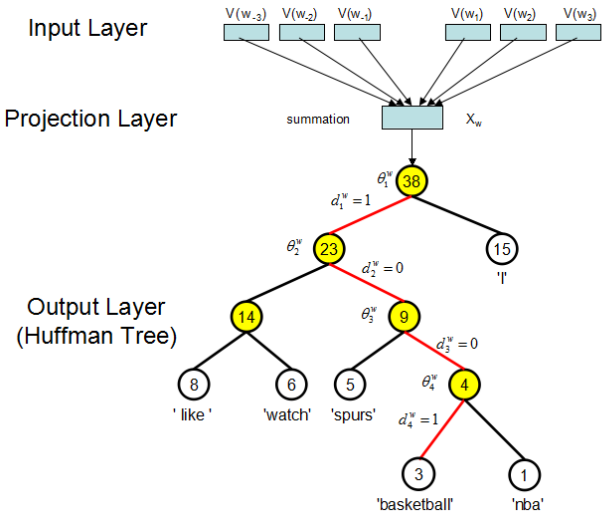

Figure 3 Detailed model structure

The basic process of this algorithm is described as below. First, count the word occurrences for each unique word in the whole corpus and build up the Huffman tree according to the word counts. Then treat the whole corpus as a very long word sequence and use a sliding window to go through the sequence step by step. As the sliding window returns 2*N*+1 word (central word w and 2*N* surrounding words) each time, use equations (1) to update the auxiliary vectors on the path p^w(that is θ_n^w) and use equation (2) to update the vectors representing the surrounding words:

$$\theta_n^{w(i+1)} = \theta_n^{w(i)} + \eta[1 - d_n^w - \sigma(X_w^T \theta_n^w)]X_w^T \quad (1)$$

$$V(w_j)^{(i+1)} = V(w_j)^{(i)} + \eta \sum_{n=1}^{l^w-1}[1 - d_n^w - \sigma(X_w^T \theta_n^w)]\theta_n^w \quad (2)$$

where θ_n^w is the auxiliary weight vectors stored at each non-leaf node. d_n^w is 1 if the path is from father node to its left node and 0 otherwise. X_w is the summation of input word vector while η is the learning rate.

3.2 Document vectors

Turning word vectors into document vectors, we try several methods. To begin with, we simply calculate the average vector for all the words in a particular document and regard the average vector as the document vector for this particular document:

$$V(D_n) = \frac{1}{l^{D_n}} \sum_{w \epsilon D_n} V(w) \quad (3)$$

where D_n is the nth document in the corpus, l^{D_n} is the length of D_n and w represents each word in D_n.

The inspiration behind this method is simple: every word vector contributes some values to the document vector. Then naturally, we consider that not every word is equally important, it is admitted that some words are more important in a particular document. Considering evaluating word importance in documents, tf-idf [14] is the first choice. So the second method we use is a weighted average based on tf-idf value of the words:

$$V(D_n) = \sum_{w \epsilon D_n} t(w) \cdot V(w) \quad (4)$$

where $t(w)$ is the tf-idf value for word *w*.

Instead of achieving document vector from word vectors, Doc2Vec is another way to maintain document vector directly by training a modified model. We implement the same structure as Doc2Vec [6]. In the initialize period, each document is regarded as a special 'word' and its document length is regarded as 'word counts' when building the corresponding Huffman tree. During the iteration process, an extra path from the root of the tree to the corresponding document leaf node is established. Addition to update the word vectors, document vectors (in fact the special 'word' vector) is also updated in the same way:

$$V(D_j)^{(i+1)} = V(D_j)^{(i)} + \eta \sum_{n=1}^{l^w-1}[1 - d_n^w - \sigma(X_w^T \theta_n^w)]\theta_n^w \quad (15)$$

We can explain this as through the iteration process, each of the 2*N* word vectors in this document has some influence on this document's document vector. The document vector is trained along with the word vectors thus should perform better for document representation.

Several researchers have showed that Word2Vec and Doc2Vec models performed very well on the syntactic tasks, but poorly on the semantic ones [7, 12]. So we also implement a model using results from Latent Dirichlet Allocation (LDA) as the pre-trained word vectors, since LDA is good at digging up latent semantic relations between documents and words. We collect one word's distribution among all the topics as the fixed length word vector for that word. We only update the document vector through the learning process and the LDA word vectors are fixed. After learning the document vectors for all the documents in the corpus, we use them as input for a supervised classifier.

3.3 Classifiers

The positive and negative documents are divided into two parts: one for training and another for testing. We use several standard classifiers to train and test our method. K Nearest neighbors (KNN) is the first classifier we use. There are two types of KNN clustering-based classifying algorithms: 1) Find all the neighbors within a certain distance or 2) find K nearest neighbors. We adopt the second method. Document vectors are spread in the 100 dimension space. For each test document, we find its K (in our case, 10) nearest neighbors in the vector space and give the test document the same label as the majority of the K documents. Then we try to use a decision tree as the classifier. The decision tree is a simple but effective learning method which takes specified attributes variables and outputs target values. In our model, the input variables are document vectors while the target variable can take a binary value (1 for advertising articles and 0 for non-advertising ones) so that the decision tree becomes a classification tree. Further, we implement a random forest classifier. Random forest is an ensemble learning method for classification which constructs multiple decision trees and combines the outputs from individual trees to achieve a better result. To classify a test document in the form of an input vector, put the vector down each of the decision trees in the forest. Each decision tree gives a classification result called the 'votes' for that class and we chooses the class with most 'votes' to be our classifying result. Finally, we use Support Vector Machine (SVM) classifier to test our method. SVM is one of the most successful supervised learning models that analyze data and recognize patterns. It filters the data so that only limited 'support' document vectors are considered when calculating a classifying hyper plane. Choosing kernel functions are what matters in SVM models. Due to the simplicity of our data set right now, we find linear kernel for SVM classifier already works well.

4 Experiments and results

4.1 Datasets

We tested our method on three data sets, Wikipedia user page dataset and Wikipedia simulate ads injection dataset created by ourselves and another one is Farm Ads Data Set collected by Mesterharm et al [13].

Table 1 Basic statistics of datasets

Wikipedia user page dataset	
Article Number	1893
Vocabulary size	26329
Average length	347 words
Positive articles	Negative articles
1470	432
Farm Ads dataset	
Article Number	4143
Vocabulary size	26839
Average length	427 words
Positive articles	Negative articles
2210	1933
Wikipedia simulate ads injection dataset	
Article Number	2470
Vocabulary size	95240
Average length	954 words
Positive articles	Negative articles
1470	1000

The Wikipedia user page dataset is labeled as two classes, one positive class represents the advertising articles and the other negative class represents the non-ad articles. In the Wikipedia logs, all deletion logs of advertising articles are labeled with a sign: 'G11: Unambiguous advertising or promotion'. We targeted at the Wikipedia user pages, since advertising user pages differ significantly from normal user pages, thus are relatively easy to recognize. From 2014 May to August, we collected 1470 deleted user pages from Google Cache as our positive documents in the dataset. As turn for the negative documents, we collected 423 Wikipedia administrators' user pages assuming that the administrators' user pages are of good qualities and satisfying Wikipedia's neutral rules. An example of two representative deleted advertising article (the upper part) and non-advertising article (the bottom part) are shown in Figure 4.

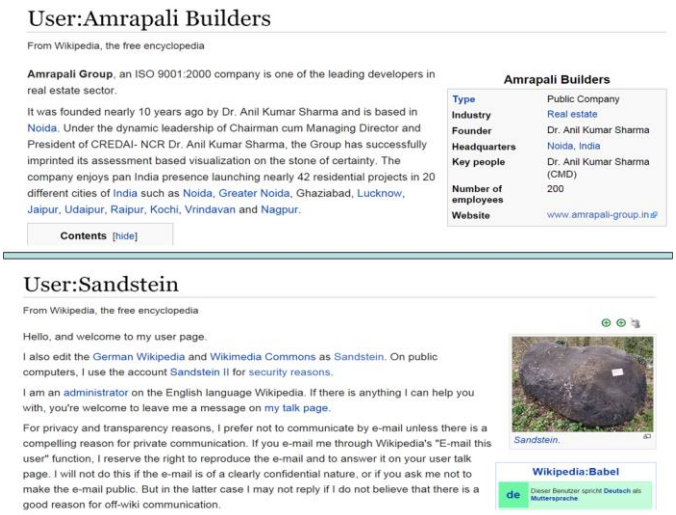

Figure 4 Examples of Wikipedia user page articles

The Wikipedia simulate ads injection dataset is created to simulate the injection of advertise into Wikipedia articles. We use 1470 deleted advertising articles as injection ads to find articles in the related areas in Wikipedia and add the advertising articles at the end of the related articles. This is based on the assumption that some advertisers will find Wikipedia articles related to their ads and modify the article page inserting their ads. Aiming to detect such kind of advertising, we create 1470 simulated advertising articles as positive samples (ad) and randomly pick 1000 Wikipedia articles as negative samples (non-ad). An example of ads injected article is shown in Figure 5.

Manufacturing Manufacturing is the production of merchandise for use or sale using labour and chemical and biological or The term may refer to a range of human from handicraft to high but is most commonly applied to industrial in which raw materials are transformed into finished goods on a large Such finished goods may be used for... ... co ltd is surrey british columbia canada based manufacturing company that designs builds rugged robust control systems marine industrial industries ...

Figure 5 Wikipedia Ads injection dataset example article

The Farm Ads data set [13] was collected from text ads found on twelve websites that deal with various farm animal related topics. Information from the ad creative and the ad landing page is included. The binary labels (1 for accepted ads and -1 for rejected ads) are based on whether or not the content owner approves of the ad. An example of two representative ad article (the upper part) and non-ad article (the bottom part) after stemming are shown in Figure 6.

No.107: pet veterinary question health care information online vet answer dog cat bird fish hamster gerbil ferret reptile horse rabbit farm animal servername www register login help veterinary veterinary question answer asap watch video pet veterinary category legal tax legal family law immigration law employment law criminal law military law real estate law canada law personal injury law business law consumer protection law estate law bankruptcy law australia law intellectual property law south africa law zealand law republic ireland law uk law uk family law uk property law uk immigration law scot law uk employment law uk bankruptcy law uk traffic law tax finance uk tax financial software car vehicle car bmw motorcycle merce vw volvo audi classic car jaguar subaru rv australia car kia porsche hyundai mitsubishi mazda saab boat marine electronics chrysler dodge jeep ford mercury lincoln gmc chevy buick pontiac cadillac saturn honda acura toyota lexus uk car uk ford uk nissan nissan infiniti heavy equipment health medical health dental mental health pharmacy medical ob gyn pediatrics urology eye dermatology oncology neurology bariatrics plastic surgery pet veterinary pet dog cat bird reptile horse management veterinary dog veterinary cat veterinary bird veterinary horse veterinary animal veterinary home appliance home improvement .....

No.3870: american fertility female infertility skip content navigation search search site search navigation service testimonial finance resource faq female infertility female infertility male infertility egg donor program advance laparoscopic surgery service blood test blood test step woman infertility easy identify potential fertility relate test doctor hormone level test follicle stimulate hormone fsh measure fsh essential start infertility evaluation prior treatment fsh level indirectly measure store follicle ovarian reserve egg oocyte remain ovary predict quality remain oocyte fsh level indicate diminish quality quantity oocyte help predict fertility doctor measure fsh level draw blood third day menstrual cycle fsh test fall range normal microiu borderline microiu abnormal microiu borderline result suggest poor ovarian reserve prompt aggressive treatment abnormal fsh level suggest poor ovarian reserve markedly low chance healthy pregnancy own egg please note fsh lab range vary institution institution estradiol estradiol type estrogen hormone produce follicle ovary elevate level third day cycle indicate compromise ovarian reserve despite normal fsh level progesterone progesterone major hormone prepare sustain uterus pregnancy produce corpus luteum embryo develop placenta blood test perform determine proper function corpus luteum.....

Figure 6 Farm Ads dataset example articles

4.2 Results and Evaluation

Experiments were conducted on the Wikipedia user page dataset and Farm Ads dataset. Different classifiers are used to test our method. As the Farm Ads dataset is more balanced, more experiments are done on it. All the experiments are done on an Ubuntu 32bit system with 2GB memory based on VMware Workstation.

Table 2 shows the experiment results on Wikipedia user page dataset. We used 1000 positive and 1000 negative articles to train the model and use the rest articles as the test set. BOW represents for Bag of Words model such that each document is coded into a fixed length (as for this Wikipedia dataset, 26329) vector and if one particular word exists in this document, the corresponding position in this document's vector is 1, otherwise 0. W2V (mean) means that we first train the model to obtain word vectors for each unique word in the corpus. Then we take average of all the word vectors of one document as its document vector. As we can see, simple BOW model already achieved very good results on this dataset. The W2V (mean) method performed better on the KNN classifier and achieved the same results on the Random Forest classifier. But the result became on the Decision Tree and SVM classifiers. This is due to the low similarity between advertising articles and non-advertising articles. Since this dataset are limited to Wikipedia user pages, frequent words in the positive documents rarely appear in the negative documents. This makes Bag of Words perfect for handling the classification task on these documents. But the BOW model suffers from the curse of dimension problem. As the dataset gradually grows, the size of the document vectors becomes larger, making the computation process really expensive. However, W2V does not have such a scalability problem so we can expect it to be better than the BOW model when we deal with larger and more complex dataset.

Table 2 Wikipedia user page dataset experiment results

Method	Precision	Recall	F1-score
BOW + KNN	0.94	0.93	0.93
BOW + Decision Tree	**0.96**	**0.96**	**0.96**
BOW + Random Forest	0.93	0.92	0.92
BOW + SVM	**0.96**	**0.96**	**0.96**
W2V (mean) + KNN	**0.96**	**0.96**	**0.96**
W2V (mean) + Decision Tree	0.90	0.90	0.90
W2V (mean)+ Random Forest	0.93	0.92	0.92
W2V (mean)+ SVM	0.81	0.82	0.81

Table 3 shows the experiment results on Wikipedia ads injection dataset. We used 1000 positive and 500 negative articles to train the model and use the rest articles as the test set. As we can see, simple BOW model works worse in this dataset comparing to the previous Wikipedia user page dataset. This is due to the increased word appearance complexity of the ads injected article as the injected ads may contain same set of words appearing in the related article which makes BOW model hard to distinguish between ads and non-ads. The

W2V model achieved equivalent results which show the effectiveness of deep learning model. When applying the D2V model, we use pre-trained word vectors[1] getting from Wikipedia 2014 and Gigaword5 dataset (around 6 billion words) since deep learning method couldn't achieve relatively good results with only several million words. The results with D2V model clearly outperform the BOW model.

Table 3 Wikipedia ads injection dataset experiment results

Method	Precision	Recall	F1-score
BOW + SVM	0.83	0.83	0.83
W2V (mean) + SVM	0.82	0.82	0.82
D2V (pre-trained) + SVM	**0.86**	**0.86**	**0.86**

Table 4 shows the experiment results on the Farm Ads dataset. We used 1000 positive and 1000 negative articles to train the model and use the rest articles as the test set. BOW represents for Bag of Words model and W2V (mean) means that we used average word vectors to represent document vectors. D2V stands for the Document to Vector model which directly learns the document vectors along with the word vectors. D2V + LDA means that we used word distributions among topics from training a 1000 pass LDA model as the word vectors in the Doc2Vec model and fix the word vectors during the D2V training process. From the results of Table 2, we can clearly tell that our method is better than the baseline BOW model. Compared to the Wikipedia user page dataset, the Farm Ads dataset is much more complex. Same words can appear both in the ad and nonad articles, making the simple BOW model inefficient. Surprisingly, the tf-idf weighted strategy to obtain document vectors from word vectors did not work well. The reason behind this is that when training the word vector, the term frequency information has already been taken into consider (using word counts to create the Huffman tree). So if we apply tf-idf weights again, it is like using the term frequency twice thus creating some imbalance leading to bad results. The D2V method shows no obvious improvements on the results but works better on the precision part. If we care more about the precision of the classifier, we should adopt the D2V structure. Adding LDA word distributions makes the results a little worse. One possible reason is that W2V training process is like clustering syntactic similar words together, while LDA word distributions are placing semantic similar words together in the vector space. Comparing to random initialization, this may even larger the distance between syntactic similar words thus making the result worse.

Table 4 Farm Ads dataset experiment results

Method	Precision	Recall	F1-score
BOW + Decision Tree	**0.81**	**0.81**	**0.81**
BOW + Random Forest	0.84	0.84	0.84
BOW + SVM	0.82	0.81	0.82
W2V (mean) + KNN	**0.85**	**0.84**	**0.84**
W2V (mean) + Decision Tree	0.80	0.80	0.80
W2V (mean)+ Random Forest	**0.84**	**0.84**	**0.84**
W2V (mean)+ SVM	**0.84**	**0.83**	**0.84**
W2V (tf-idf) + SVM	0.75	0.46	0.32
D2V + SVM	**0.85**	**0.82**	**0.82**
D2V + LDA + SVM	**0.82**	**0.82**	**0.82**

[1] http://nlp.stanford.edu/projects/glove/

5 Conclusion and future work

In this paper, we described a method to automatically detect advertising articles in Wikipedia. We first implement a deep learning method to either directly obtain document vectors or from the learned word vectors. Then the document vectors in the training dataset are used to train a supervised classifier which does the classification for the test dataset in the next step. Experiment results based on two dataset showed that our method achieves more than 0.8 precision and recall scores and works better than the baseline bag of words model.

In the future, more work need to be done. We plan to extend our dataset to the whole Wikipedia range, detect all the potential advertising articles using our method and check the performance by keeping track of the Wikipedia deletion logs. We also plan to try other methods to improve the initial part of the Word2Vec model which now takes random values. Applying new classifiers and new models are also under our considerations.

References

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin: A Neural Probabilistic Language Model, Journal of Machine Learning Research 3 (2003) , Pages 1137–1155 (2003)

[2] Chris Dyer: Notes on Noise Contrastive Estimation and Negative Sampling, arXiv:1410.8251 [cs.LG] (2014)

[3] Yoav Goldberg, Omer Levy: word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method

[4] Kelly Y. Itakura, Charles L.A. Clarke: Using Dynamic Markov Compression to Detect Vandalism in the Wikipedia. SIGIR '09 Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, Pages 822-823 (2009)

[5] Ho-yu Lam, Dit-yan Yeung: A learning approach to spam detection based on social networks. Proceedings of the fourth conference on email and anti-spam (2007)

[6] Quoc Le, Tomas Mikolov: Distributed Representations of Sentences and Documents, JMLR '14 Proceedings of the 31st International Conference on Machine Learning, W&CP volume 32 (2014)

[7] Yang Liu, Zhiyuan Liu1, Tat-Seng Chua, Maosong Sun: Topical Word Embeddings, AAAI '15 Association for the

Advancement of Artificial (2015)

[8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean: Distributed Representations of Words and Phrases and their Compositionality, NIPS '13, Pages 3111–3119 (2013)

[9] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean: Efficient Estimation of Word Representations in Vector Space, ICLR '13 Proceedings of Workshop at International Conference on Learning Representations (2013)

[10] Martin Potthast, Benno Stein, Robert Gerling: Automatic Vandalism Detection in Wikipedia, ECIR '08 Proceedings of the 30th European Conference on IR Research (2008)

[11] Lin Qiu, Yong Cao, Zaiqing Nie, Yong Rui: Learning Word Representation Considering Proximity and Ambiguity, AAAI '14 Association for the Advancement of Artificial (2014)

[12] Radim Řehůřek: Making sense of word2vec, http://radimrehurek.com/2014/12/making-sense-of-word2vec/

[13] Chris Mesterharm, Michael J. Pazzani: Active Learning using On-line Algorithms, KDD ’11 Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, Pages 850-858, 2011

[14] Karen Spärck Jones: A statistical interpretation of term specificity and its application in retrieval, Document retrieval systems, Pages 132-142