SEMANTIC DOCUMENT CLASSIFICATION BASED ON STRATEGIES OF SEMANTIC SIMILARITY COMPUTATION AND CORRELATION ANALYSIS

Shuo Yang[1*], Ran Wei[2], Hengliang Tan[1], and Jiao Du[1]

[1]School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, China
[2]Department of Computer Science, University of California, Irvine, California, USA

ABSTRACT

Document (text) classification is a common method in e-business, facilitating users in the tasks such as document collection, analysis, categorization and storage. Semantic analysis can help to improve the performance of document classification. Though having been considered when designing previous methods for automatic document classification, more focus should be given to semantics with the increase number of content-rich electronic documents, forum posts or blogs online, which can reduce human workload by a great margin. This paper proposes a novel semantic document classification approach aiming to resolve two types of semantic problems: (1) polysemy problem, by using a novel semantic similarity computing strategy (SSC) and (2) synonym problem, by proposing a novel strong correlation analysis method (SCM). Experiments show that our strategies can help to improve the performance of the baseline methods.

KEYWORDS

semantic document classification, semantic similarity, semantic embedding, correlation analysis, machine learning

1. INTRODUCTION

Automatic document classification has many applications in numerous electronic business (e-business) scenarios [2]. For example, a medium-sized company may receive quite a few emails daily without accurate and concrete information such as recipient's name or department, which have to be read by an assigned agent so that the destinations can be determined. Thus, it is no doubt that an automatic document classification system can reduce human workload to a great extent.

More generally, given the rapid growth of web digital documents, it is often beyond one's ability to categorize information by reading thoroughly the pool of documents. Accurate and automatic text classification techniques are hence needed, which can classify the incoming text documents into different categories such as news, emails, contracts, reports, etc. Users can hence estimate the content and determine the priorities of each document, maintaining more organized working schedule and creating more business value [28].

A quantitative definition of text classification was proposed by Aggarwal and Zhai [1]: given a set of text documents $D = \{x_1, x_2, ... x_N\}$, each document x_i has to be assigned with a set of different selected indices $\{1, 2, ..., k\}$ that represents k different labels of text categories from an overall index list.

A typical method of automatic text classification is that given a training set of documents with known category labels and word dependency information, calculation on each member of the test document set has to end up with a list of possibilities on each label assigned to it. Certainly, the label with the highest likelihood corresponds to the predicted category that a test document belongs to. Classical machine learning (ML) algorithms such as Bayesian classifier, decision Tree, K-nearest neighbour, support vector machine and neural network were often applied in text classification [16]. In recent years deep learning algorithms are also introduced in these tasks. One of the representative trials was the application of convolutional neural network (CNN), a powerful network in computer vision [17]. Recurrent neural network, which has memory function that can capture sequence-formed information, was later introduced and became popular to handle classification problems [36].

However, most baselines mentioned above seldom view the classification problem from the perspective of semantic analysis. For example, the traditional Bayesian-based text classification method constructs a classification model based on the frequencies of some feature words in corpus. Unfortunately, this method does not take into account polysemous words (a word which holds different meanings depending on the context) and synonymous words (different words which hold a similar meaning) for semantic analysis during the classification procedure. For example, the Chinese word “Xiaomi” can mean either an agricultural product or a high-tech company; therefore, when classifying documents based on the traditional Bayesian method, documents including “Xiaomi” may be classified as “agriculture” or “technology”. Similar problems also exist in the classification of English documents. For example, English documents containing the word “program” may not only represent computer code programs and be classified as “computer”, but also represent a scheduled radio or television show and be classified as “entertainment”. Similarly, English articles containing the word “center” can either represent a geometric center and be classified as “mathematics”, or an important place of economy and culture and be classified as “geography”.

On the other hand, synonymous words can also cause mis-classification of documents. For example, the word “people” is synonymous with “mass” and “mob”. But they may occur in articles about different topics (e.g., architecture, culture and history). Therefore, choosing these words as features of the classification model may cause classification errors. These situations also exist in document classification tasks of word-embedding-based deep learning methods. For example, during feature extraction procedure the word dependence is calculated based on network training upon a particular corpus; in other words, the result is based on the statistical analysis on the posterior probability of a word following another one. However, a single embedding cannot represent multiple meanings, while similar embeddings may refer to different topic types.

The above issues can be summarized as two research problems:

(1) Problem of polysemy: some words have multiple meanings, which may lead to mis-classification of documents;

(2) Problem of synonym: different words with similar meanings are often used in different scenarios, but when they appear in an article at the same time, it may lead to mis-classification of documents;

Khan et al. [16] suggested that semantic analysis could help enhance the performance of classification. Previous work has made significant progress on this task. Fang, Guo, Wang and Yang (2007), and Khan, Baharudin, Lee and Khan (2010) claimed that semantic analysis can be generally implemented by the introduction of ontology that represents terms and concepts in a domain-wised manner [8,16]. However, ontologies are particularly pre-defined domain-constraint expert knowledge base. They are not good at eliminating ambiguity across different fields (domains) or different natural languages, which may lead to polysemy and synonymy issues [29], finally resulting in uncertainty of document classification [9]. Liu, Scheuermann, Li and Zhu (2007) proposed a text classification method based on WordNet for word sense disambiguation (WSD) [20]. Some other approaches use supervised (Jin, Zhang, Chen, Xia (2016) [12]) or unsupervised method or the joint method of them (Wawer and Mykowiecka (2017) [32]) for word disambiguation. However, few methods consider document misclassification caused by both the ambiguity of polysemy and multi-scene characteristics of synonym at the same time. In recent years, some approaches use name entities for text classification. For example, Stefan, Miroslav, Ivan, Marko and Aleksandar (2017) proposed a method based on name entity network linking. However, the author showed that their experiment results did not show any significant improvement when using named entities, and in some cases even worse performance [28]. Türker, Koutraki, Zhang and Sack (2018) proposed an approach based on a name entity dictionary (i.e., Anchor-Text Dictionary). However, if the words of the text do not exist in the dictionary, the classification results may be biased [30].

HIT IR-Lab Tongyici Cilin (Extended) proved that extending word meaning effectively or replacing keywords with synonyms can significantly improve the performance of information retrieval, text classification and automatic question answering system [13]. Motivated by this linguistic evidence, in this research, we propose two strategies to improve the performance of semantic document categorization of baselines. The *first* strategy aims to solve polysemy problem by using a novel semantic similarity computing method (SSC) so that the most context-fitting meaning of a word can be determined by referring to the meaning of similar sentences in a common dictionary. In this paper, *CoDic* [10,34] and *Hownet* [7] are used as common dictionaries for meaning determination and term expansion. With the help of CoDic and Hownet, words with ambiguity will be removed from the feature list, enabling more distinctive features to be selected. The *second* strategy aims to solve the synonym problem by adopting a strong correlation analysis method (SCM), where synonyms unrelated to the classification task are deleted. Otherwise, select the specific meaning of one word in the synonym group from the common dictionary and replace the other synonyms in the same group.

2. RELATED WORK

Automated document classification, also called categorization of document, has a history that can date back to the beginning of the 1960s. The incredible increase in online documents in the last decades intensified and renewed the interests in automated document classification and data mining. In the beginning, document classification focused on heuristic methods, that is, solving the task by applying a group of rules based on expert knowledge. However, this method was proved to be inefficient, so in recent years more focuses are turned to automatic learning and clustering approaches. These approaches can be divided into three categories based the characteristics of their learning phases:

(1) *Supervised document classification*: this method guidelines the whole learning process of classifier model by providing complete training dataset that contains document content and category labels at the same time. The process of supervision is like that of students doing exercises which have correct answers for them to refer to.

(2) *Semi-supervised document classification*: a mixture method between supervised and unsupervised document classification. Parts of documents have category labels while the others do not.

(3) *Unsupervised document classification*: this method is executed without priori knowledge of the document categories. The process of unsupervised learning is like that of students doing final examination which they do not have standard answers for reference.

However, no matter what kinds of learning methods, many of them require to firstly convert unstructured text to digital numbers in the data pre-processing stage. The most traditional (and intuitional) algorithm is one-hot representation, which uses N-dimension binary vector to represent vocabulary with each dimension stands for one word [16]. However, this strategy easily incurs curse of dimensionality for representation of long texts. This is because a big vocabulary generates high-dimension, but extremely sparse vectors for long documents. Therefore, dimensionality reduction operation which removes redundant and irrelevant features is needed [4]. This demand is satisfied by the methodology called feature extraction/selection. The goal of feature extraction is the division of a sentence into meaningful clusters and meanwhile removing insignificant components as much as possible. Typical tasks at the pre-processing stage include tokenization, filtering, lemmatization and stemming [31]. After that, feature selection aims to select useful features of a word for further analysis. Compared with one-hot representation that generates high-dimensional, sparse vectors, an improved solution called TF-IDF produces more refined results. In this frequency-based algorithm, the importance of a word is represented by the product of term frequency (how frequent the word shows up in a document) and inverse document frequency (log-inverse of the frequency that documents containing such word in the overall document base) [21,31]. These two algorithms, however, clearly suffer from limitations brought by neglecting the grammar and word relations in documents. More recently, distributed representation that illustrates dependencies between words are more widely used, as it reflects the relationships of words in one document [23]. Currently, the most widely used strategy to learn the vectorized words is to maximize the corpus likelihood (prediction-based), with the word2vec toolbox being one of the most popular tools. Implementation of this algorithm is dependent on the training of representation neural network with words in the form of binary vectors generated by one-hot representation. The weights of the network keep being updated until convergence, which generates a vector that lists the possibility of each word could follow the input word in a document [16,23].

3. SEMANTIC DOCUMENT CLASSIFICATION

This section proposes two novel strategies to resolve the research problems mentioned above.

3.1. Strategy to Resolve Polysemy Problem: SSC

The first strategy aims to solve polysemy problem by using a novel semantic similarity computing method. The most context-fitting meaning of a word can be determined by referring to the semantics of related sentences in a common dictionary (e.g., CoDic for English and Hownet for Chinese).

In this strategy, we implement the semantic similarity computing method (*SSC*) for the similarity between two sentences. The *SSC* splits a text document into sentences. For each word (*w*) in a sentence (*s*), all of its concepts from the dictionary are extracted based on its Part-of-speech (*PoS*) tag in the sentence. Then, semantically compare each concept of *w* with *s* and return the concept with the maximum similarity score. Words that are not determinative of their exact meanings will be removed from the list of features, and hereby more distinctive terms are more likely to be selected as features. The pseudocode of the *SSC* algorithm is shown as Table 1.

The workflow of the *SSC* is quite simple. From Table 1, it is clear that the first step is to segment each sentence into words (*word_tokenize*) and tokenize each word (*pos_tag*) with its part of speech. Then, we get the synonym set (*synset*) for each tagged word in the sentence according to their PoS (*tagged_to_synset*). After that, we filter out the null component in each synset. Next, for each synset in the first sentence (*sent1*), we compute the similarity score of the most similar word (*compute_similarity*) in the second sentence (*sent2*). The aim of our function *compute_similarity* is to measure the similarity between two synsets. If two words are similar, their synsets should also be similar. This is because if two words are very similar, then their correlations with the same some other words will be very close. On the other hand, if the correlation between two words and the same some other words is close, then the two words are similar to each other [26].

Table 1. Semantic similarity computing (SSC)

Algorithm: semantic similarity computing (SSC)
Input: target sentence (ts); a set of test sentences (ss)
Output: the most similar sentence (s in ss) to ts with its maximum similar score (max)

```
def sentence_similarity (sentence1, sentence2)
#Tokenize & pos tag
   sentence1 = pos_tag(word_tokenize(sentence1))
   sentence2 = pos_tag(word tokenize(sentence2))
# Get the synsets for the tagged words
   synsets1 = [tagged_to_synset(*tagged word) for tagged_word in sentence1]
   synsets2 = [tagged_to_synset(*tagged word) for tagged_word in sentence2]
# Filter out the Null values
   synsets1 = [synset1 for synset1 in synsets1 if synset1]
```

```
    synsets2 = [synset2 for synset2 in synsets2 if synset2]
    score, count = 0.0, 0
 # For each word in the first sentence
 for synset1 in synsets1
 # Get the similarity score of the most similar word in the second sentence
    best_score = max([synset1.compute_similarity(synset2) for synset2 in synsets2] )
 # Check that whether the similarity could have been computed if best score is not None
    score += best_score
    count += 1
 # Average the values
    score /= count
    return score # end of sentence similarity function
 # __main__
    max = 0.0
    most_similar_sentence = None
 for s in ss
    value1 = sentence_similarity(s, ts)
    value2 = sentence_similarity(ts, s)
    avg_similarity = (value1 + value2) / 2
    if avg_similarity >maximum:
       most_similar_sentence = s
       max = avg_similarity
 print (“The most similar sentence is {}, with score {}”.format(most similar sentence, max))
```

In the function of *compute_similarity*, when calculating the similarity of any two words in two synsets, we applied the mean value of multiple methods (if applicable): Path Similarity (PS) [3], Leacock-Chodorow (LCH) [18], Wu-Palmer (WUP) [33] and Lin [19]. This is because when using thesaurus (dictionary) alone to calculate the similarity, if the word is not in the dictionary, the similarity cannot be calculated.

PS computes the shortest number of edges from one word to another, assuming that a hierarchical structure exits (like WordNet that is essentially a graph) [22]. In general, two word that have a longer path distance are less similar than those with a very short path distance. If there is no path between two words, PS will return a Null value. This is another reason why we use different similarity measures.

$$\mathrm{sim}_{\mathrm{path}}(\mathrm{c}_1, \mathrm{c}_2) = \mathrm{pathLen}(\mathrm{c}_1, \mathrm{c}_2) \quad (1)$$

where c_1, c_2 are two words, and pathLen(c_1,c_2) is the shortest number of edges between those two words in a given thesaurus.

LCH is almost the same as PS, except it uses the negative logarithm of the result of the length of path.

$$\mathrm{sim}_{\mathrm{path}}(\mathrm{c}_1, \mathrm{c}_2) = -\log(\mathrm{pathLen}(\mathrm{c}_1, \mathrm{c}_2)) \quad (2)$$

Based on LCH, WUP metric expands it by weighting the edges according to the distance in the hierarchy. Unlike the above methods, Lin metric considers similarity as both the information content shared between two words, and the difference. It calculates the probability of the lowest common word between two words c_1 and c_2, which is the lowest-leveled node in the hierarchy that is the parent of both c_1 and c_2 based on the corpora used [22].

After computing the similarity score of all synsets of sent1 with that of sent2, an average similarity value between them can be returned. By using this method, we can acquire the similarity values between all test sentences (ss) and the target sentence (ts). In the end, the test sentence with the maximum similarity value can be chosen as the most semantically similar sentence.

3.2. Strategy to Resolve Synonym Problem: SCM

There may be many synonyms in a large text, but not all of them are suitable as text features. As is known to all, selecting effective text features can reduce the dimension of feature space, enhance the generalization ability of the model and reduce overfitting, so as to improve the effect and efficiency of classification and clustering [5]. Therefore, effective feature selection is particularly important. In this section, we can turn the synonym problem into a sub-problem: how to determine the degree of the relevance between a feature and the classification task and then remove the feature words in the synonym group that are weakly relevant to or irrelevant to the classification task.

In this paper, a novel correlation analysis algorithm, named SCM, is proposed to obtain effective feature sets. The idea of the SCM contains two important considerations:

The feature words with strong category discrimination ability are extracted by using the category discrimination method (CDM), and then the correlation between other feature words and categories is measured by the feature correlation analysis (FCA). That is, the selected feature is guaranteed to be the most relevant to the category first, and then the degree of correlation between other features and selected features is calculated.

If a feature has a strong correlation with the selected feature, the SCM will not include it into the feature candidate set even if the feature has a strong correlation with the category. Because compared with existing feature candidate set, the new undetermined features cannot provide additional category-related information.

This paper adopts TF-IDF (Term Frequency-Inverse Document Frequency) [14,27] as the implementation of CDM. By applying TF-IDF to the synonym group in undetermined features, we can get a feature candidate set composed of a number of features with strong category discrimination ability. The TF-IDF method is a frequency-based algorithm. In TF-IDF, the importance of a word is represented by the product of the word frequency (i.e., the frequency with which the word appears in the document) and the inverse document frequency (i.e., dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient). The formulas of TFIDF are as follows.

$$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{3}$$

$$\mathrm{idf}_i = lg\frac{|D|}{|\{D_j: t_i \epsilon\, d_j\}| + 1} \tag{4}$$

$$\mathrm{tf} - \mathrm{idf}_{i,j} = tf_{i,j} * idf_i \tag{5}$$

where (3) refers to the importance of a term t_i in a particular document d_j. The molecule $n_{i,j}$ is the number of occurrences of t_i in d_j, and the denominator is the sum of the number of occurrences of all words in d_j. Formula (4) is a measurement of the general importance of a word in all documents. Its molecule represents the total number of documents in the corpus. The denominator represents the number of documents containing the word t_i. Formula (5) is the product of "term (word) frequency (TF)" and "inverse document frequency (IDF)". The more important a word is to a certain category of texts, the higher its tf-idf value will be, and vice versa. Therefore, TF-IDF tends to filter out common words and retain important words to certain category of texts.

The SCM proceeds to calculate how strongly all features (in each synonym group) are related to category (C) in the feature candidate set. The formulas are as follows,

$$\mathrm{H(x)} = \sum_{i=0}^{n}(p_i * lg\frac{1}{p_i}) \tag{6}$$

$$H(X|Y) = \sum_j p(Y_j) \sum_i p(X_I|Y_j) lg\frac{1}{P(X_i|Y_j)} \tag{7}$$

$$I(X|Y\,) \;=\; H(X) \;-\; H(X|Y\,) \tag{8}$$

$$\mathrm{Corr(X,Y)} = \frac{I(X|Y) + I(Y|X)}{H(X) + H(Y)} \tag{9}$$

where X is an n-dimensional random variable and Y is a certain of class (or category). Formula (6) represents the entropy of X, that is the uncertainty of X. Formula (7) means the uncertainty of X given the occurrence of Y. Formula (8) represents information gain between $H(X)$ and $H(X|Y\,)$. Formula (9) is used to measure the degree of correlation between a feature (X) and a category (Y).

According to the degree of correlation, the features in each synonym group are arranged in a descending order respectively, and then the ordered feature sequences are put back into the feature candidate set. Select the first feature in the sequence, that is, the feature with the strongest correlation with category (C), and remove it from the feature candidate set and put it into the feature result set.

In order to eliminate redundant features, it is necessary to calculate the degree of mutual independence between any two features (within a synonym group). Thus, this section proposes a novel feature correlation analysis method, called FCA, to exclude unnecessary features in synonym groups of the feature candidate set. The idea of the FCA is simple: if a remaining feature in the candidate set is a strong category-correlated feature, and its mutual independence with the selected feature is greater than or equal to a threshold *alpha*, it indicates that the candidate feature is independent of the selected feature, and it needs to be included in the feature result set. Otherwise, the feature is considered as redundant and should be deleted. Repeat this process until the feature candidate set is empty. The formulas are as follows:

$$\mathrm{IDP}(\mathrm{X}_i, \mathrm{Y}|\mathrm{X}_j) = \frac{I(\mathrm{X}_i, \mathrm{Y}|\mathrm{X}_j) + I(\mathrm{X}_j, \mathrm{Y}|\mathrm{X}_i)}{2H(Y)} \tag{10}$$

$$\mathrm{I}(\mathrm{X}; \mathrm{Y}|\mathrm{Z}) = \lg \frac{p(X|YZ)}{p(X|Z)} \tag{11}$$

where (10) is used to measure the degree of mutual independence between feature X_i and feature X_j when the category (Y) is known. Formula (11) describe the mutual information between feature X and feature Y in the case of given condition Z.

4. EXPERIMENTS

This section first introduces the datasets and evaluation metrics. Then, we experiment our strategies based on several baselines with detailed experimental procedure. After that, classification assessment is given based on the performance.

4.1. Dataset and Evaluation Metrics

To test the reliability and robustness of our strategy, we use:

Dataset 1: a movie review dataset from Rotten Tomatoes [24, 37]. This dataset contains 10662 samples of review sentences, with 50% positive comments and the remaining negative ones. The size of the vocabulary of the dataset is 18758. Since the dataset does not come with an official train/test split, we simply extract 10% of shuffled data as evaluation (dev) set to control the complexity of model. In the next research stage, we will use 10-fold cross-validation on the dataset.

Dataset 2: 56821 Chinese news dataset, which is available in PaddlePaddle [1] that is an open source platform launched by Baidu for deep learning applications. It contains 10 categories: international (4354), culture (5110), entertainment (6043), sports (4818), finance (7432), automobile (7469), education (8066), technology (6017), stock (3654) and real estate (3858).
We assess the classification quality automatically with macro-average on accuracy and loss.

4.2. Experiment on neural network (NN)

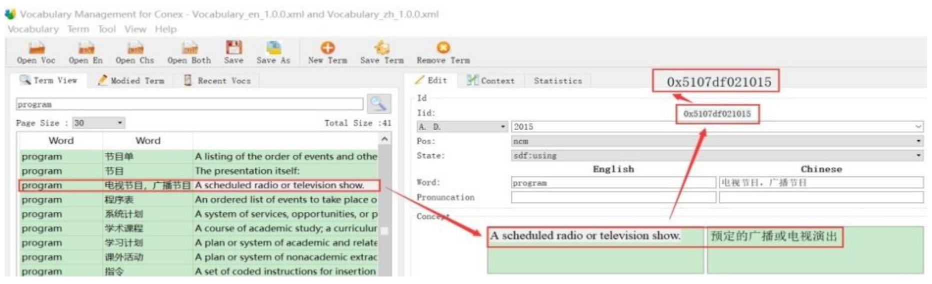
In this experiment, the baseline CNN is taken as an example to compare the performance of classical NN and the improved one with our proposed strategy in document classification. The detail of model parameters is listed in Table 2. Both of the two trained models are evaluated on the *dev* dataset every 100 global steps and then they are stored in checkpoints before the training process starting again. After multiple training epochs, the models stored in checkpoint can be recovered and used for testing on a new dataset. Partial code for this work is available on github [2]. The experimental procedure is described as follows.

(1) Each document in the corpus will be firstly transformed into our semantic document (i.e., documents with semantics embedding) [35] by extending each polysemous word and category-correlated synonymous word with its context-fitting concepts from the common dictionary (i.e., CoDic for English and Hownet for Chinese) with the help of the SSC and the SCM strategies, which aims for accurate semantic interpretation and term expansion.

CoDic is a semantic collaboration dictionary constructed under our CONEX project [10,34,35]. In CoDic, each concept is identified by a unique internal identifier (iid). The reason of this design is to guarantee semantic consistency and interoperability of documents while transferring across heterogeneous contexts. For example, from Figs. 1 and 2, it is clear that in CoDic, the word “program” with the meaning of “a scheduled radio or television show” is uniquely labelled by an iid “0x5107df021015”, while its another meaning “a set of coded instructions for insertion into a machine...” has another unique iid “0x5107df02101c”. Currently, CoDic is implemented in XML, where each concept is represented as an entry with a unique *iid* (see Fig. 3). It is convenient to extract all different meanings of any given word for later semantic analysis by using existed packages (e.g., *xml.etree.cElementTree* for Python and *javax.xml.parsers* for Java). Hownet as a common dictionary to handle Chinese documents is used similarly.

Table 2. Parameter settings of our experiments

Parameters	**values**
Percentage of splitting a dataset for training, testing and validating, respectively	0.8/ 0.1/ 0.1
Dimensionality of character embedding	128
Filter sizes	3,4,5
Number of filters per filter size	128
Dropout keep probability	0.5
L2 regularization lambda	0.01
Batch Size	64
Number of training epochs	1/ 5/ 10/ 50/ 100
Evaluate model on evaluation (dev) dataset after these steps	100



program

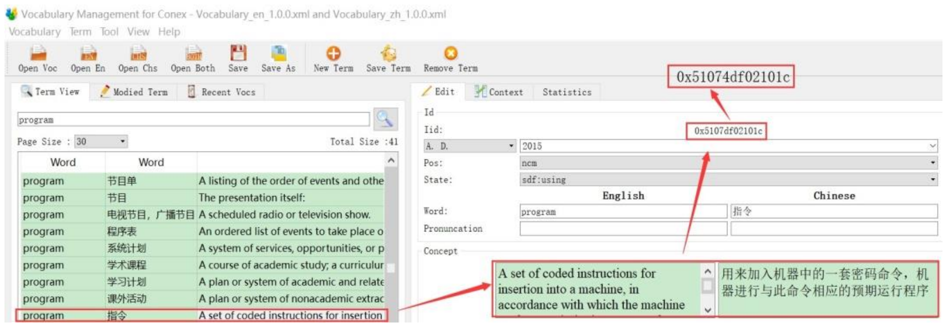
Page Size : 30 Total Size :41

Word	Word	
program	节目单	A listing of the order of events and othe
program	节目	The presentation itself:
program	电视节目，广播节目	A scheduled radio or television show.
program	程序表	An ordered list of events to take place o
program	系统计划	A system of services, opportunities, or p
program	学术课程	A course of academic study; a curriculur
program	学习计划	A plan or system of academic and relate
program	课外活动	A plan or system of nonacademic extrac
program	指令	A set of coded instructions for insertion

Id
Iid: 0x5107df021015
A. D. 2015
Pos: ncm
State: sdf:using

	English	Chinese
Word:	program	电视节目，广播节目
Pronuncation		
Concept	A scheduled radio or television show.	预定的广播或电视演出

Figure 1. Word “program” with the meaning “a scheduled radio or television show” in CoDic


Vocabulary Management for Conex - Vocabulary_en_1.0.0.xml and Vocabulary_zh_1.0.0.xml
Vocabulary Term Tool View Help
Open Voc Open En Open Chs Open Both Save Save As New Term Save Term Remove Term
Term View Modied Term Recent Vocs
Edit Context Statistics

program

Page Size : 30 Total Size :41

Word	Word	
program	节目单	A listing of the order of events and othe
program	节目	The presentation itself:
program	电视节目，广播节目	A scheduled radio or television show.
program	程序表	An ordered list of events to take place o
program	系统计划	A system of services, opportunities, or p
program	学术课程	A course of academic study; a curriculur
program	学习计划	A plan or system of academic and relate
program	课外活动	A plan or system of nonacademic extrac
program	指令	A set of coded instructions for insertion

Id
Iid: 0x5107df02101c
A. D. 2015
Pos: ncm
State: sdf:using

	English	Chinese
Word:	program	指令
Pronuncation		

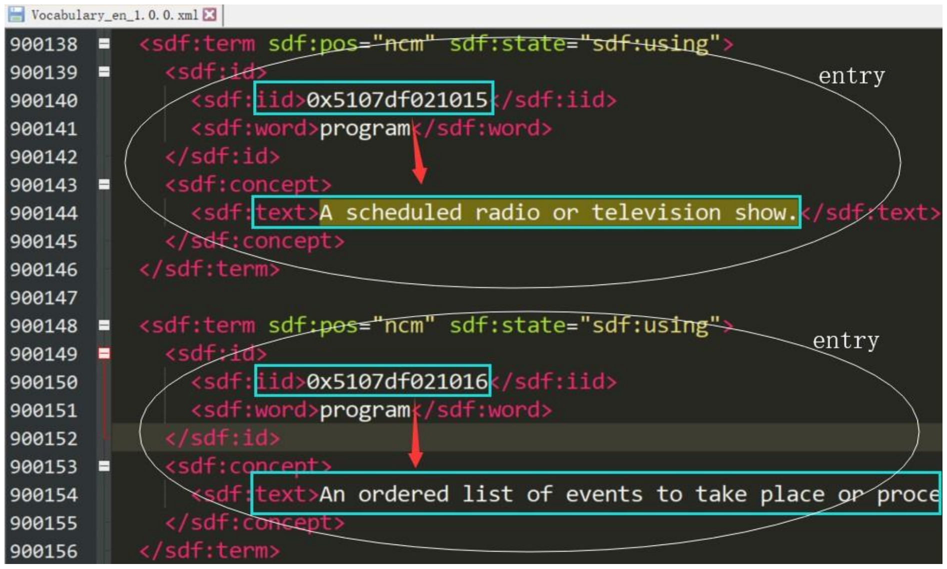
Concept

A set of coded instructions for insertion into a machine, in accordance with which the machine

用来加入机器中的一套密码命令，机器进行与此命令相应的预期运行程序

0x51074df02101c

Figure 2. Word “program” with the meaning “a set of coded instructions for insertion into a machine” in CoDic



```
<sdf:term sdf:pos="ncm" sdf:state="sdf:using">
  <sdf:id>
    <sdf:iid>0x5107df021015</sdf:iid>
    <sdf:word>program</sdf:word>
  </sdf:id>
  <sdf:concept>
    <sdf:text>A scheduled radio or television show.</sdf:text>
  </sdf:concept>
</sdf:term>

<sdf:term sdf:pos="ncm" sdf:state="sdf:using">
  <sdf:id>
    <sdf:iid>0x5107df021016</sdf:iid>
    <sdf:word>program</sdf:word>
  </sdf:id>
  <sdf:concept>
    <sdf:text>An ordered list of events to take place or proce
  </sdf:concept>
</sdf:term>
```

Figure3. CoDic in XML

(2) Build a *Sem*$_{CNN}$ (CNN+SSC/SCM) network. The first layer embeds words and their extracted accurate concepts into low-dimensional vectors. The second layer performs convolutions over the semantic-embedded document tensors using different sized filters (e.g., *filter size* = [3, 4, 5]). Different sized filters will create different shaped feature maps (i.e., tensors). Third, max-pooling is used to merge the results of the convolution layer into a long feature vector. Next, dropout regularization is added in the result of max-pooling to trade-off between the complexity of the model being trained and the generalization of testing on evaluation dataset. The last layer is to classify the result using a Softmax strategy.

(3) Calculate loss and accuracy. The general loss function for classification problems is the cross-entropy loss which takes the prediction and the real value as input. Accuracy is another useful metric being tracked during training and testing processes. It can be used to prevent model overfitting during model training. At the beginning of the training, the training error on training dataset and the verification error on the evaluation dataset will decrease continuously. However, when the training process reaches a certain critical point, the accuracy of classification on the evaluation dataset will decline while the accuracy of training will continue to increase. At this

time, in order to avoid overfitting of the model, the training process should be interrupted and the parameters at the critical point should be used as the training results of the model.

(4) Record the summaries/checkpoints during training and evaluation. After an object declaration of CNN/Sem_{CNN} class, batches of data are generated and fed into it to train a reliable classification model. While the loss and accuracy are recorded to keep track of their evolvement over iterations, some important parameters (e.g., the embedding for each word, the weights in the convolution layers) are also needed to be saved for later usage (e.g., testing on new datasets).

(5) Test the classification model. Data for testing are loaded and their true labels are extracted for computing the performance of prediction. Then, the classification model is restored from the checkpoints, executing on the test dataset and producing a prediction for each semantic document. After that, the prediction results are compared with the true labels to obtain the testing accuracy of the classification model.

4.3. Experiment on ML approaches

The procedures of training classification models using classical machine learning algorithms with the proposed strategies are listed as follows, while the details can be also found in our open source code.

(1) Transform words into vectors based on inputted texts (Note: Chinese document needs to execute word segmentation beforehand.). Collect all words used in texts, perform a frequency distribution and then find out effective features suitable for document classification by using the proposed strategies (SSC and SCM). After that, each text will be converted to a long word vector, where True (or 1) means a word (or a feature) exists while False (or 0) means absent.

(2) Execute multiple classical machine learning approaches (e.g., Naïve Bayes, NB) based on the word vectors from Step (1). In this experiment, three variants of NB classifier are used. They are Original NB, multinomial NB and Bernoulli NB classifier. All of them take word features and corresponding category labels as input to train classification models. It is of note that sometimes the classifier should be modified based on realistic cases. For example, in order to avoid the probability being close to zero and underflow problem in NB, it is better to initialize the frequency of each word to one and take natural log of the product in the computation of posterior probability, respectively.
(3) Save the trained classifiers for later usage. This is because the training process might be time-consuming, which depends on numerous factors such as dataset size and the computation complexity during model training. Thus, it is impractical to train classification models each time while you need to use them.

(4) Boost multiple classifiers to create a voting system that is taken as a baseline for comparison. To do this, we build a typical classifier (i.e., *VoteClassifier*) with multiple basic classical ML classification algorithms (i.e., taking multiple basic classifier objects as input when initialized), each of which gets one vote. In *VoteClasssifier*, the *classify* method is created by iterating through each basic ML classifier object to classify based on the same input features. This experiment chooses the most popular metrics (e.g., accuracy) among these classifiers. The classification can be regarded as a vote. After iterating all the classifier objects, it returns the most popular vote.

4.4. Experiment Result and Analysis

In the actual testing process, we need to maintain a common synonymous word dictionary and a common polysemous word dictionary. The reason we need to maintain these two dictionaries is that the computation workload to judge polysemy and synonyms in a long text are very heavy. For example, if there are n words in a text and each word has m different meanings, then the computational complexity of determining polysemous words is $O(n*m)$, and the computational complexity of determining synonyms is $O(n*(n-1))$, so that the total computational complexity is $O(n*(m+n-1)) > O(n^2)$. Therefore, maintaining these two dictionaries can reduce computational complexity and reduce the pre-processing time of text classification.

Table 3 shows the experimental comparison between classical machine learning algorithms and their improved counterparts on Dataset 1. In this experiment, classical machine learning algorithms include Original Naïve Bayes (NB), Multinomial Naïve Bayes (MNB), Bernoulli Naïve Bayes (BNB), Logistic Regression (LR), support vector machine (SVM) with stochastic gradient descent (SGD), Linear SVC (SVC) and Nu-Support Vector Classification (NSVC).

From Table 3, it is clear that our improved algorithms have better performance than the classical ML algorithms in the accuracy of model prediction on the evaluation dataset. It is of note that three-variant NB algorithms and LR perform better than three-variant SVM algorithms, in both of the classical ones and improved ones. The VoteClassifier plays a role of baseline for the comparison between different algorithms. Table 4 and 5 show that Sem_{CNN} performs better than CNN in terms of accuracy and loss in different numbers of epochs. As the number of epoch increases, both of them increase in the accuracy of evaluation and decrease in the loss continuously (**before reaching overfitting**).

Table 3. Comparison of classical machine learning algorithms and our improved ones on Dataset 1

Accuracy (%)		**Accuracy (%)**	
NB	73.493	Improved NB	78.464
MNB	74.698	Improved MNB	79.518
BNB	74.096	Improved BNB	79.819
LR	73.494	Improved LR	76.506
SGD	69.879	Improved SGD	74.096
SVC	72.741	Improved SVC	73.946
NSVC	72.892	Improved NSVC	76.355
VoteClassifier	74.397	Improved VoteClassifier	74.398

Table 4. Comparison of SemCNN and traditional CNN on Dataset 1.

Number of epochs	**Accuracy**		**Loss**	
	Sem_{CNN}	**CNN**	***Sem_{CNN}***	**CNN**
Epoch = 1	0.586	0.568	0.818	0.876
Epoch = 5	0.713	0.676	0.567	0.59
Epoch = 10	0.744	0.722	0.519	0.62
Epoch = 50	0.841	0.724	0.621	0.742
Epoch = 100	0.902	0.739	0.961	0.999

Table 5. Comparison of SemCNN and traditional CNN on Dataset 2.

Number of epochs	**Accuracy**		**Loss**	
	Sem_{CNN}	**CNN**	***Sem_{CNN}***	**CNN**
Epoch = 1	0.861	0.828	0.473	0.560
Epoch = 5	0.956	0.923	0.211	0.295
Epoch = 10	0.990	0.953	0.095	0.212
Epoch = 20	0.990	0.966	0.0919	0.170

5. CONCLUSION

This paper introduces new strategies for semantic document classification. It mainly has two improvements: (1) solving polysemy problem by using a novel semantic similarity computing method (SSC). The SSC implements semantic analysis by executing semantic similarity computation and semantic embedding with the help of common dictionary. In this paper, we use CoDic for English texts and Hownet for Chinese texts. (2) solving synonym problem by proposing a novel strong correlation analysis method (SCM). The SCM consists of the CDM strategy for the selection of feature candidate set and the FCA strategy for the determination of the final feature set. Experiments show that our strategy can improve the performance of semantic document classification compared with that of traditional ones.

We will continue going deep in this research of semantic document classification. More multiple deep learning models (e.g., DualTextCNN, DualBiLSTM, DualBiLSTMCNN or BiLSTMAttention) will be tested for semantic document similarity on well-known document datasets with different natural languages. We would also try to compare our strategies with state-of-the-art embedding methods such as FastText [15], BERT [6] and ULMFit [11] and ELMo [25] and other classification methods such as the ones based on knowledge graph.

ACKNOWLEDGMENT

This research is supported by both National Natural Science Foundation of China (grant no.: 61802079 and 61802418) and Guangzhou University grant (no.: 2900603143). The authors would like to thank all the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

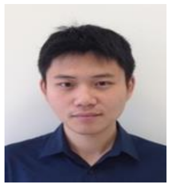
[1] Aggarwal, C.C., Zhai, C.: Mining text data. Springer Science & Business Media (2012)
[2] Altınel, B., Ganiz, M.C.: Semantic text classification: A survey of past and recent advances. Information Processing & Management 54(6), 1129–1153 (2018)
[3] Budanitsky, A., Hirst, G.: Evaluating wordnet-based measures of lexical semantic relatedness. Computational Linguistics 32(1), 13–47 (2006)
[4] Cerda, P., Varoquaux, G., K´egl, B.: Similarity encoding for learning with dirty categorical variables. Machine Learning 107(8-10), 1477–1494 (2018)
[5] Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Computers & Electrical Engineering 40(1), 16–28 (2014)
[6] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
[7] Dong, Z., Dong, Q., Hao, C.: Hownet and the computation of meaning (2006)
[8] Fang, J., Guo, L., Wang, X., Yang, N.: Ontology-based automatic classification and ranking for web documents. In: Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007). vol. 3, pp. 627–631. IEEE (2007)
[9] Gambhir, M., Gupta, V.: Recent automatic text summarization techniques: a survey. Artificial Intelligence Review 47(1), 1–66 (2017)
[10] Guo, J., Da Xu, L., Xiao, G., Gong, Z.: Improving multilingual semantic interoperation in cross-organizational enterprise systems through concept disambiguation. IEEE Transactions on Industrial Informatics 8(3), 647–658 (2012)
[11] Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146 (2018)
[12] Jin, P., Zhang, Y., Chen, X., Xia, Y.: Bag-of-embeddings for text classification. In: IJCAI. vol. 16, pp. 2824–2830 (2016)
[13] Jiu-le, T., Wei, Z.: Words similarity algorithm based on tongyici cilin in semantic web adaptive learning system [j]. Journal of Jilin University (Information Science Edition) 6(010) (2010)
[14] Jones, K.S.: A statistical interpretation of term specificity and its application in retrieval. Journal of documentation (2004)
[15] Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
[16] Khan, A., Baharudin, B., Lee, L.H., Khan, K.: A review of machine learning algorithms for text-documents classification. Journal of advances in information technology 1(1), 4–20 (2010)
[17] Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
[18] Leacock, C., Chodorow, M.: Combining local context and wordnet similarity for word sense identification. WordNet: An electronic lexical database 49(2), 265–283 (1998)
[19] Lin, D., et al.: An information-theoretic definition of similarity. In: Icml. vol. 98, pp. 296–304. Citeseer (1998)
[20] Liu, Y., Scheuermann, P., Li, X., Zhu, X.: Using wordnet to disambiguate word senses for text classification. In: international conference on computational science. pp. 781–789. Springer (2007)
[21] Manning, C.D., Raghavan, P., Schu¨tze, H.: Scoring, term weighting and the vector space model. Introduction to information retrieval 100, 2–4 (2008)
[22] Martin, J.H., Jurafsky, D.: Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. Pearson/Prentice Hall Upper Saddle River (2009)
[23] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
[24] Pang, B., Lee, L.: Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of the 43rd annual meeting on association for computational linguistics. pp. 115–124. Association for Computational Linguistics (2005)

[25] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018)
[26] Qun, L., Sujian, L.: Semantic similarity calculation based on zhiwang. International Journal of Computational Linguistics and Chinese Language Processing 7(2), 59– 76 (2002)
[27] Salton, G., Fox, E.A., Wu, H.: Extended boolean information retrieval. Tech. rep., Cornell University (1982)
[28] Stefan Aneli, Miroslav Kondi, I.P.M.J.A.K.: Text classification based on named entities. In: ICIST. pp. 23–28 (2017)
[29] Thangaraj, M., Sivakami, M.: Text classification techniques: A literature review. Interdisciplinary Journal of Information, Knowledge & Management 13 (2018)
[30] Tu¨rker, R., Zhang, L., Koutraki, M., Sack, H.: Tecne: Knowledge based text classification using network embeddings. In: EKAW (Posters & Demos). pp. 53–56 (2018)
[31] Wang, Y., Wang, X.J.: A new approach to feature selection in text classification. In: 2005 International conference on machine learning and cybernetics. vol. 6, pp. 3814–3819. IEEE (2005)
[32] Wawer, A., Mykowiecka, A.: Supervised and unsupervised word sense disambiguation on word embedding vectors of unambigous synonyms. In: Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications. pp. 120–125 (2017)
[33] Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: Proceedings of the 32nd annual meeting on Association for Computational Linguistics. pp. 133–138.
[34] Association for Computational Linguistics (1994)
[35] Xiao, G., Guo, J., Gong, Z., Li, R.: Semantic input method of chinese word senses for semantic document exchange in e-business. Journal of Industrial Information Integration 3, 31–36 (2016)
[36] Yang, S., Wei, R., Shigarov, A.: Semantic interoperability for electronic business through a novel cross-context semantic document exchange approach. In: Proceedings of the ACM Symposium on Document Engineering 2018. p. 28. ACM (2018)
[37] Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent trends in deep learning based natural language processing. ieee Computational intelligenCe magazine 13(3), 55–75 (2018)
[38] Zhang, Y., Wallace, B.: A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820 (2015)

AUTHORS

Shuo Yang received the Master's degree in software engineering from the Dalian Jiaotong University, China, in 2013. He was awarded a doctorate degree in software engineering, University of Macau, in 2017. Currently, he is a researcher in Guangzhou University. His research interests include semantic interoperability and semantic inference with AI technology, mainly applied to the fields of e-commerce, e-marketplace and clinical area.

Ran Wei received the Ph.D. degree in biomedical science from Rutgers University, USA, in 2018. He is currently a researcher in the Department of Computer Science, University of California, Irvine. His interests focus on bioinformatics, health informatics and artificial intelligence-aided healthcare.



Jiao Du was born in Chongqing, P.R.China, in1988. She received M.S. and Ph. D degree from Chongqing University of Posts and Telecommunications, Chongqing, P.R.China in 2013 and 2017, respectively. Currently, she is a lecturer with the school the School of Computer Science and Educational Software, Guangzhou University, Guangzhou 510006, China. Her research interests include pattern recognition and image fusion.

Hengliang Tan received his B.E. degree from Foshan University, Foshan, China, in 2006 and his M.E. and Ph.D. degrees from Sun Yat-sen University, Guangzhou, China, in 2011 and 2016, respectively. He joined the School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, China in 2016. His current research interests include machine learning, pattern recognition and manifold learning.