2014 IEEE 26th International Conference on Tools with Artificial Intelligence

Text Summarization Based on Sentence Selection with Semantic Representation

Chi Zhang, Lei Zhang, Chong-Jun Wang, Jun-Yuan Xie
National Key Laboratory for Novel Software Technology,
Nanjing University
Nanjing 210023, China
Email: zc83354965@gmail.com, zhanglei.com@gmail.com, {chjwang,jyxie}@nju.edu.cn

***Abstract*—Text summarization is of great importance to solve information overload. Salience and coverage are two most important issues for summaries. Most existing models extract summaries by selecting the top sentences with highest scores without using the relationships between sentences, and usually represent the sentences simply basing on lexical or statistical features. As a result, those models can not achieve salience or coverage very well. In this paper, we propose a novel summarization model called Sentence Selection with Semantic Representation (SSSR). SSSR ensures both salience and coverage by learning semantic representations for sentences and applying a well-designed selection strategy to select summary sentences. The selection strategy used in SSSR is to select sentences that can reconstruct the original document with least distortion by means of linear combination. Besides, we improve our selection strategy by reducing redundant information. Then we learn two semantic representations for sentences: (1) weighted mean of word embeddings; (2) deep coding. Both of them are semantic and compact, and can capture similarities between sentences. Extensive experiments on datasets DUC2006 and DUC2007 validate our model.**

***Keywords*-document summarization, selection strategy, sentence representation, lasso, word2vec**

I. Introduction

In recent years, with the development of social media and user-generated content, the problem of information overload has attracted increasing attention. Automatic text summarization is of great importance to solve this problem.

Generating a concise and fluent summary requires the capability to reorganize, modify and merge information expressed in different sentences in the input. All the methods of automatic text summarization could be divided into two categories: extraction-based and abstraction-based. Extractive summaries (extracts) are produced by concatenating several sentences taken exactly as they appear in the materials being summarized. In contrast, abstractive summaries (abstracts) are produced by building an internal semantic representation and then creating summaries using natural language generation techniques. Abstracts are usually closer to human generated summaries, however, abstractive summarization is known to be very difficult. The state-of-the-art methods in eliciting semantic representations and natural language generation are still very weak. Therefore, most researchers focus their attention on extractive summarization, which is also our research focus on in this paper.

Salience and coverage are two important issues for evaluating the quality of extractive summaries[1]. A salient summary should contain the sentences that are most relevant to the main topics of original document. Besides, a good summary is expected to cover as many topics as possible and has mininum redundancy.

There have been a variety of studies in automatic extraction, but their performances are less than satisfactory. Compared with human summary, computer-generated summary suffers from lower quality and coverage. Most previous summarization models estimate the salience of a sentence using predefined features, such as term frequency, lexical chains[2], word co-occurrence[3] and centrality[4]. However, these lexical or statistical features can not effectively capture semantic meanings. From the perspective of physiology, a well-performing automatic summarization should provide human-like judgment by referencing human's neocortex and the procedure of intelligent perception. Therefore, it is important to learn deeper level semantic features to represent sentences and document.

On the other hand, most existing summarization approaches are based on calculating salience scores for each sentence and composing the result summary of the top sentences with highest scores. These ranking based models usually score sentences independently without using the relationships between sentences. Therefore, ranking based models are unable to model coverage explicitly. Coverage is the second most important property after salience and it can be achieved by applying sentence selection strategies, e.g. maximal marginal relevance (MMR)[5].

In this paper, we propose a novel summarization model called Sentence Selection with Semantic Representation (SSSR). In contrast with previous ranking based models and lexical feature based models, SSSR can ensure both salience and coverage of the generated summary by learning semantic sentence representations and applying a well-designed selection strategy.

SSSR consists of two parts: sentence selection strategy and sentence representation learning. The selection strategy used in SSSR is to select the sentences that can best reconstruct the original document by means of linear combination. Considering the length limit on summary, we add sparsity constraints on the combination coefficients. Using

1082-3409/14 $31.00 © 2014 IEEE
DOI 10.1109/ICTAI.2014.93

ℓ_1 penalty to encourage sparsity, the optimization problem can be turned into Lasso. By minimizing the cost, we get the optimal sentence set as the result summary. Besides, we also improve our selection strategy by reducing redundant information.

Before applying the selection strategy, semantic representations of sentences and document are essential. In this paper, SSSR learns two representations for sentences: (1) weighted mean of word embeddings; (2) deep coding. Both of them are expressive and compact, and able to capture the similarities between sentences.

The remaining parts of this paper are organized as follows. Section 2 provides a brief review of previous extraction approaches. Section 3 describes the selection strategy used in SSSR in detail. Section 4 introduces two approaches to learn semantic representations for sentences. Section 5 gives the experimental results and the final section concludes this paper.

II. Related Work

Extraction approaches can be categorized as supervised and unsupervised approaches. Supervised approaches consider the summarization as a two-class classification problem where sentences are either labelled as in-summary or not-in-summary. Discriminative models such as Support Vector Machine (SVM)[6] classify each sentence individually. Structured output models such as Hidden Markov Model (HMM)[7] and Conditional Random Field (CRF)[8] are then proposed to take into account relationships between sentences.

Most existing unsupervised summarization methods are based on sentence ranking. These ranking models obtain the top sentences with highest scores as the result summary. The scoring criterias of ranking models include lexical or statistical features such as lexical chains[2] and word co-occurrences[3]. Inspired by PageRank algorithm, graph based models such as LexRank[4] score sentences by centrality.

These ranking models focus on estimating the salience of sentences, whereas selection models focus on coverage. Carbonell[5] proposed famous maximal marginal relevance (MMR) which uses a greedy algorithm to select sentences by reducing redundancy. Inspired by Latent Semantic Analysis (LSA), Gong[9] uses singular value decomposition (SVD) to select sentences for generic document summarization. He[10] proposes document summarization based on data reconstruction (DSDR) model that selects sentences by reconstructing all the sentences in document with non-negative linear functions.

III. Sentence Selection Strategy

A. Document Reconstruction

Inspired by He[10], our selection strategy is to select sentences that can best reconstruct the original document with least distortion. The difference is that He reconstructs all the original sentences in document and obtains the cost function by accumulating the reconstruction error of each sentence. However our selection strategy represents the entire document as a vector and attempts to simply reconstruct the document vector.

We first represent each sentence and the entire document individually as a feature vector, of which the detailed learning procedure will be explained in Section IV. Then we donate the sentence set as matrix $X = \{s_1, s_2..\}$, where each column s_i stands for the representation of the i_{th} sentence in document. The document is represented as vector y with the same dimensionality as s_i.

Given the sentence set X, SSSR attempts to reconstruct the document y using a reconstruction function. In this paper, we use linear combination as the reconstruction function and squared error as the reconstruction loss:

$$\|y - Xw\|_{\ell_2}^2 \tag{1}$$

where w stands for the combination coefficient vector.

The sentences corresponding to zero coefficients in w are considered as irrelevant sentences, because they are not involved in document reconstruction. As a result, summary sentences will be selected from the sentences that correspond to the non-zero coefficients in w.

There is always a length limit on the result summary. For example, the summarization test system we conduct experiments on restricts the generated summary of each topic to 250 words. As a result, we would like the combination coefficients to have as few non-zero components as possible. Sparsity constraints should be added to the coefficients. ℓ_0 cardinality is the most intuitive measure of sparsity. Using ℓ_0 penalty, the objective function is obtained as:

$$\min_w \|y - Xw\|_{\ell_2}^2 + \lambda \|w\|_{\ell_0} \tag{2}$$

where λ is a parameter that controls the trade-off between sparsity and reconstruction fidelity.

By minimizing the cost function, we get the optimal coefficient vector w. The sentences corresponding to the non-zero coefficients in w will be selected to compose the result summary. Least Squares Estimate (LSE) with ℓ_0 regularization is known as Sparse Coding. In view of Sparse Coding, sentence matrix X is a set of over-complete bases, document y is an input vector to be compressed, and w is the output coding under the over-complete basis with a sparsity restriction.

However the ℓ_0 penalty is non-convex and difficult to optimize. The common solution is to replace the ℓ_0 penalty with ℓ_1 or ℓ_2 penalty. Replacing the ℓ_0 penalty in Eq. 2 with ℓ_2 yields the ridge regression, which is an continuously differentiable convex optimization problem. It brings an advantage over original LSE that ℓ_2 penalty gives a compromise between solving the system and having a small

w. While ℓ_2 penalty is an effective regularization means of achieving numerical stability and variables shrinkage, it does not strongly encourage sparsity, and the resulting models typically have non-zero values associated with all coefficients. It is proved that ℓ_1 processes sparsity better and often outperforms ℓ_2 penalty when irrelevant features exist in matrix X[11]. Given the above, we use ℓ_1 regularization as the sparsity penalty. The objective function can be then obtained as :

$$\min_{w} \|y - Xw\|_{\ell_2}^2 + \lambda \|w\|_{\ell_1} \tag{3}$$

LSE with ℓ_1 penalty was presented and popularized independently under the names Least Absolute Selection and Shrinkage Operator (LASSO)[12] and Basis Pursuit Denoising (BPDN)[13]. Lasso is a typical instance of convex optimization and quadratic programming. There have been a wide variety of approaches proposed for the Lasso problem. The Least Angle Regression (LARS)[14] algorithm gives an efficient way of solving Lasso and connects the Lasso to forward stagewise regression. Coordinate descent[15] is extremely simple and fast, and exploits the assumed sparsity of the model to great advantage.

λ is a parameter that controls the trade-off between sparsity and reconstruction fidelity. Regression models usually use cross validation to obtain the optimal λ for precise predictions, while our selection strategy is to obtain the optimal coefficient vector w for any given λ. By manually adjusting the value of λ, we can get the result summary of desired length.

Data pre-processing is an indispensable step, including data centralization and normalization. During the preprocessing, columns of sentence matrix X are standardized to have zero-mean and unit-norm, while document vector y is centered to zero-mean. By centering the columns of X and y, we remove the intercept. By normalizing the columns of X, the columns are all put on the same scale and the coefficients are of the same magnitude, in case that a column with a very high norm gets a very low coefficient.

B. Redundancy Reduction

We also improve our selection strategy by reducing redundancy. Redundancy is defined as overlapping information between sentences. Abstractive summaries can reduce redundancy by sentence fusion, whereas it is very difficult for extraction models to generated non-redundant summaries.

The main challenge of non-redundant extraction is that redundancy is always calculated between the candidate sentence and the sentences already included in the summary, which depends on using a greedy algorithm to select sentence by sentence, e.g. MMR. Besides, over-reduction of redundancy might lead to information distortion.

The objective function Eq. (3) might come up with negative coefficients and their corresponding sentences are then considered to be redundant sentences. To minimize the redundant information, we add non-negative constraints on the combination coefficients.

Lasso with a non-negative constraint is defined as Positive Lasso and it can be solved by LARS modification[14]. By optimizing the Positive Lasso, we obtain a non-negative coefficient vector w, and the sentences corresponding to the non-zero coefficients in w will be selected to compose the result summary.

IV. Sentence Representation Learning

Before applying the selection strategy described in the previous section, SSSR learns semantic representations of sentences and documents. The most common text representation is bag of words (BOW). BOW is a simplifying representation and widely used in information retrieval and document classification. In this representation, a piece of text (a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and word order but keeping term frequency. The word bag is represented as a vector, where each dimension refers to a separate term weighted by its frequency. BOW vectors suffer from high dimensionality and sparsity, and can not effectively capture the semantic meanings. However, BOW is simple and effective to be used as an approximate representation in some cases. In latter deep coding learning, we will use BOW vectors as the inputs of autoencoder.

To learn semantic and low-dimensional representations of sentences, we use two representations in this paper: weighted mean of word embeddings and deep coding.

A. Weighted Mean of Word Embeddings

Word embedding is a vector mapped from a word to represent it. One-hot representation is a simple form of word embedding, which represents each word as a bit vector. However its dimensionality is vocabulary-size and it can not capture semantic word relationships. In this paper, SSSR uses another form of word embedding known as distributed representation. Distributed representation was first proposed by Hinton[16]. Hinton describes distributed representation as "a relation between two types of representation". In this form, all words are mapped into the same feature space each as a low-dimensional vector. In the feature space, distance metric such as Euclidean distance or cosine of angle can be used to calculate the semantic similarity between words. Distributed representation is proved to have better performances in NLP tasks by grouping similar words[17].

Distributed word embeddings are usually learned by neural networks. Bengio[18] first learned a distributed representation for words via a neural probabilistic language model, which is originally used to compute the joint probabilities of word sequences. Among the follow-up studies, word2vec[19] toolkit developed by Google provides state-of-the-art word embeddings.

Word2vec provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for training vector representations of words. These vectors perform well in many NLP tasks such as word clustering and machine translation. The most popular property of word2vec is that the word vectors capture many linguistic regularities, for example, *vector('king') - vector('man') + vector('woman') = vector('queen')*. In this paper, SSSR uses word2vec tool to learn vector representations of words and each produced vector is 300-dimensional.

We presume that the text (a document or sentence) representation is some kind of weighted mean of the word embeddings in the text. Under this assumption, we first represent the text as an unordered word set that contains no duplicate elements $\{w_1, w_2, \cdots\}$. Each word element is assigned its word embedding produced by word2vec. Then we calculate the weighted mean of word set as the sentence representation:

$$s = \frac{\sum_{i=1}^{n} weight(w_i) * embedding(w_i)}{\sum_{i=1}^{n} weight(w_i)} \tag{4}$$

where $weight(w_i)$ is a weighting function that captures the importance of word w_i in the sentence. In this paper we use tf-idf as the weighting function.

Neuroscience shows different neocortex areas together with dozens of cortical layers are involved in generating even the simplest lexical-semantic processing. Therefore, deep architecture is identical to the multi-layer physical structure of the human cerebral cortex[20]. In this paper, we train a deep autoencoder to learn low-dimensional and semantic codings of sentences, whereas word2vec, in contrast, uses a shallow neural network.

Deep autoencoder was first proposed by Hinton[21] for handwriting recognition and document retrieval. The deep architecture achieves dimensionality reduction using an adaptive neural network, which includes a multilayer "encoder" network to learn compact codings from the high-dimensional inputs and a similar "decoder" network to reconstruct the original data from the codings. To avoid local minimum and gradient diffusion caused by random initialization, pretraining via a stack of Restricted Boltzmann machines (RBM) is necessary and efficient.

B. Deep Coding

In this paper, we build a 1000-900-300-100-30 autoencoder to learn 30-dimensional codings of sentences. To train this autoencoder, we choose 10000 sentences picked from New York Times and Associated Press, and each sentence sample is represented as a bag-of-words vector of 1000 highest-frequency word stems. The training procedure includes three stages: pretraining, unrolling and fine-tuning. In pretraining stage, two adjacent layers form a undirected bipartite graph and it can be trained efficiently by using a RBM to learn one layer of hidden variables at a time.

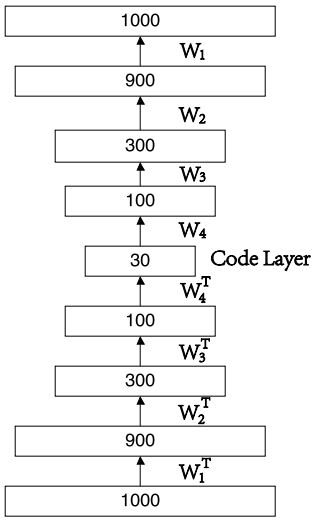


Figure 1: After pretraining, RBMs are unrolled to create a 1000-900-300-100-30 autoencoder that is fine-tuned by backpropagation. This deep autoencoder extracts a 30-dimensional coding for each input sentence.

Each RBM has only one layer of feature detectors and its learned feature activations are treated as the input data of the next RBM in the stack. After the greedy layer-wise pretraining, we unroll these four RBMs and get a complete deep structure. Then the network parameters are fine-tuned by minimizing cross entropy error with back-propagation.

Note that word counts aren't quite the same as pixels or real values, the bag-of-words vectors are often divided by document size to convert the input vector into a probability vector, allowing the deep autoencoder behave sensibly when dealing with documents of different lengths. However the probabilities are so small that the input units would have very small activities. To solve this problem, Hinton scales up the up-going weights of the bottom RBM by document size and then normalizes the outputs using a softmax function[21]. While Salakhutdinov models the bag-of-words vectors using a constrained Poisson model[22]. In this paper, we simply rescale the range of input probability vectors as:

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{5}$$

where x is an input probability vector, $max(x)$ stands for the maximum value in vector x and $min(x)$ stands for the minimum. This min-max-scaling makes the probabilities multiply and can still scale their range in [0,1] without using softmax normalization.

V. Experimental Results

A. Dataset and Evaluation Metric

The Document Understanding Conference (DUC) has been carrying out large-scale evaluations of summarization

systems on common datasets since 2001. Over 20 teams participate in this NIST-run evaluation each year and lots of efforts have been invested by the conference organizers to improve evaluation methods. To verify the summarization performance of SSSR, we conduct experiments on DUC2006 and DUC2007 datasets, which are commonly used in text summarization evaluation tasks. DUC2006 and DUC2007 datasets contain 50 and 45 topics respectively, with 25 news articles in each topic. Besides, each topic offers 4 human summaries wrote by NIST assessors for evaluation. For each topic, the generated summary is restricted to 250 words.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)[23] is the most popular evaluation tool for text summarization. ROUGE is based on computing overlapping units such as n-gram, word sequences and word pairs between the peer summary produced by participants and the model summary produced by assessors. Basically, ROUGE-N measure compares N-grams of two summaries, and counts the number of matches. Rouge-N score is computed as:

$$ROUGE-N = \frac{\sum_{S \in Ref} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in Ref} \sum_{gram_n \in S} Count(gram_n)}$$

where Ref stands for the set of model summaries. $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in peer summary and model summaries, and $Count(gram_n)$ is the number of n-grams in model summaries.

Besides Rouge-N, Rouge-L uses the longest common sequence (LCS) metric and Rouge-SU4 scores by the overlaps of skip-bigrams with unigram scoring for a maximum skip distance of four. In this paper, we provide the results of the average F-measure scores based on Rouge-1, Rouge-2, Rouge-3, Rouge-L and Rouge-SU4 on both DUC2006 and DUC2007 datasets.

To validate the effectiveness of SSSR, We compare our experimental results with other representative summarization algorithms, including a ranking model: Self-Present Sentence Relevance Model (SPSR)[24], two selection models: DSDR and TopicDSDR[25], and two baselines offered by NIST.

These compared methods are described briefly as follows:

- **Baseline1**: for each topic, Baseline1 simply takes the leading sentence of each news in the document set.
- **Baseline2**: in DUC2007, NIST offers a new baseline using an automatic summary system $CLASSY04$.
- **SPSR**: scores sentences by minimizing the latent document implicit relevance structure and then selects top-k highest-scored sentences.
- **DSDR**: selects summary sentences by reconstructing all the sentences in original document.
- **TopicDSDR**: is an improved model of DSDR. TopicDSDR represents each sentence as a topic vector using Latent Dirichlet Allocation (LDA).

B. Results Comparison

The F-measure scores on five ROUGE metrics are shown in our experimental results: Rouge-1, Rouge-2, Rouge-3, Rouge-L and Rouge-SU4. Table 1 and Table 2 show the ROUGE evaluation results on DUC2006 and DUC2007 data sets respectively. "SSSR-w" and "SSSR-d" respectively donate SSSR model with weighted mean of word embeddings representation and deep coding representation.

Table I: Average F-measure performance on DUC2006. "SSSR-w" and "SSSR-d" donate SSSR with weighted mean of word embeddings representation and deep coding representation respectively.

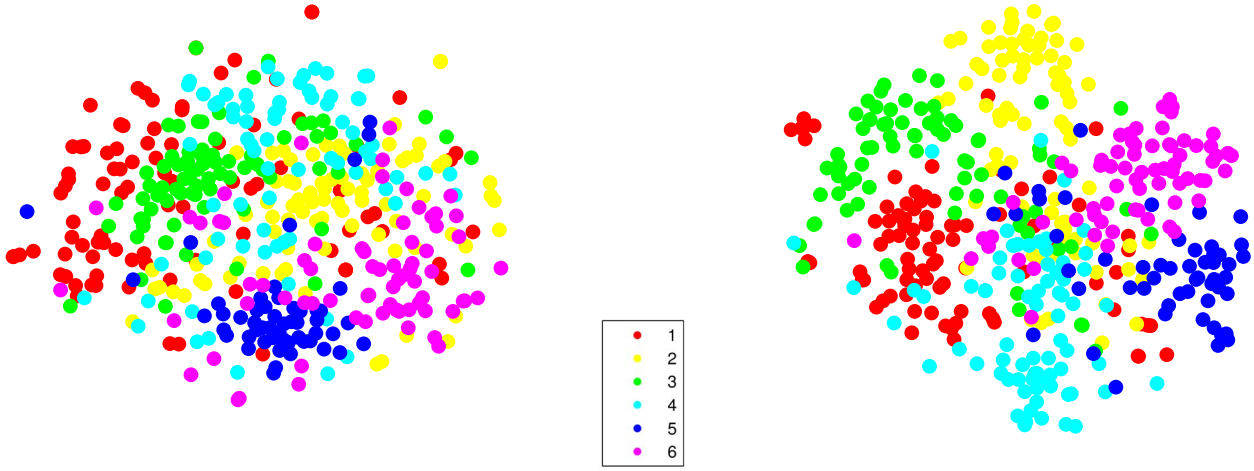
Method	Rouge-1	Rouge-2	Rouge-3	Rouge-L	Rouge-SU4
Baseline1	0.32082	0.05267	0.01372	0.29726	0.10408
SPSR	0.35383	0.05922	0.01555	0.31354	-
DSDR	0.33168	0.06047	0.01482	0.29850	-
TopicDSDR	0.37635	0.07073	-	0.34172	**0.13190**
SSSR-w	0.38365	**0.07142**	**0.02001**	0.34113	0.12944
SSSR-d	**0.38463**	0.06826	0.01494	**0.34785**	0.12831

Table II: Average F-measure performance on DUC2007. "SSSR-w" and "SSSR-d" donate SSSR with weighted mean of word embeddings representation and deep coding representation respectively.

Method	Rouge-1	Rouge-2	Rouge-3	Rouge-L	Rouge-SU4
Baseline1	0.33475	0.06490	0.01856	0.31074	0.11278
Baseline2	0.40059	0.09272	0.03058	0.36317	0.14467
SPSR	0.37070	0.06716	0.01844	0.32704	-
DSDR	0.39573	0.07439	0.01965	0.35335	-
TopicDSDR	0.39849	0.08200	-	0.36164	0.14562
SSSR-w	**0.41431**	0.09558	0.03106	0.37355	0.14861
SSSR-d	0.41124	**0.09588**	**0.03152**	**0.38636**	**0.15061**

As is shown by the Rouge scores in two tables, SSSR takes the lead followed by TopicDSDR model. Baseline1 method that selects the leading sentences performs very poor, whereas Baselines2 performs well, benefiting from the effectiveness of $CLASSY04$. SPSR takes into account the relationships between sentences, but the sentences are represented as nonsemantic bag-of-word vectors. TopicDSDR uses the same selection strategy as DSDR, but represents sentences as topic vectors using LDA, which can capture semantic sentence meanings. As a result, TopicDSDR obviously outperforms original DSDR.

Under the SSSR framework, SSSR with deep coding representation slightly outperforms SSSR with weighted mean of word embeddings representation. To prove that deep coding representation is more semantic and effective than weighted mean of word embeddings representation, we conduct the following experiment.



(a) weighted mean of word embeddings representation

(b) deep coding representation

Figure 2: Representations of all sentences in the six news picked from DUC2007, each dot stands for a different sentence, and each color stands for a different news. Two sentence representations both have good performances in sentence classification. Deep coding representation performs slightly better.

C. Sentence Representations Comparison

To compare the two sentence representations, we conduct sentence classification experiments on six news picked from DUC2007, including *"An Interview with BURMA's AungSan Suu Kyi", "Gingrich Wraps up His Official Duties", "Spain Facing its Own Fears on Separatism", "Search for Suspected Abortion Clinic Bomber Still Wide Open", "E-Commerce: Coming Soon to A Coffee Shop Near You", "Napster Strikes Dissonant Chords"*. Each news has about 100 sentences. We visualize the representations of all sentences produced by both approaches to see their classification performances, using t-SNE tool [26], which is a widely used tool for visualizing high dimensional data. The result visualization is shown in Figure 2. The figure shows that both approaches perform well in sentence classification, which means they can both learn compact and semantic representations.

At the same time, we can also see that deep coding representation slightly outperforms weighted mean of word embeddings representation in the sentence classification task. This suggests that deep coding representation learns more effective features of sentences, which we believe is due to the use of deep architecture. However, the weighted mean of word embeddings representation can be computed very efficiently, it remains an interesting research question to find a different weighting scheme (other than the tf-idf in Eq. 4) that can reach a balance between efficiency and effectiveness.

VI. Conclusion and Future Work

In this paper, we propose a novel summarization method called Sentence Selection with Semantic Representation (SSSR). SSSR ensures both salience and coverage by learning semantic representations of sentences and applying a well-designed selection strategy to select summary sentences. The selection strategy is to select sentences that can best reconstruct the original document by means of linear combination to compose the result summary. We also introduce two representations for sentences, including weighted mean of word embeddings and deep coding, which are both semantic and compact. The experimental results validate the capability and effectiveness of SSSR.

There are still challenges ahead to follow-up on this work. We list some works that we intend to explore later:

- The selection strategy applied in SSSR uses a linear reconstruction function, which is not so flexible as non-linear function. For further exploration, non-linear functions could be employed and analyzed to select sentences.
- Recently there are some researches on learning representations for sentences by neural networks, especially recurrent neural networks. These sentence representations could be applied in SSSR to improve the summarization performance.

Acknowledgements

The authors would like to thank anonymous reviewers for their helpful comments, Jian-Xin Wu for his useful feedbacks. This paper is supported by the National Natural Science Foundation of China(Grant No.61375069, 61105069)

and the Science and Technology Support Foundation of Jiangsu Province(Grant No. BE2012161).

REFERENCES

[1] Y. Ouyang, W. Li, R. Zhang, S. Li, and Q. Lu, "A progressive sentence selection strategy for document summarization," *Information Processing & Management*, vol. 49, no. 1, pp. 213–221, 2013.

[2] R. Barzilay and M. Elhadad, "Using lexical chains for text summarization," *Advances in automatic text summarization*, pp. 111–121, 1999.

[3] Y. Matsuo and M. Ishizuka, "Keyword extraction from a single document using word co-occurrence statistical information," *International Journal on Artificial Intelligence Tools*, vol. 13, no. 01, pp. 157–169, 2004.

[4] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *J. Artif. Intell. Res.(JAIR)*, vol. 22, no. 1, pp. 457–479, 2004.

[5] J. Carbonell and J. Goldstein, "The use of mmr, diversity-based reranking for reordering documents and producing summaries," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1998, pp. 335–336.

[6] J.-Y. Yeh, H.-R. Ke, W.-P. Yang, I. Meng *et al.*, "Text summarization using a trainable summarizer and latent semantic analysis," *Information Processing & Management*, vol. 41, no. 1, pp. 75–95, 2005.

[7] J. M. Conroy and D. P. O'leary, "Text summarization via hidden markov models," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2001, pp. 406–407.

[8] D. Shen, J.-T. Sun, H. Li, Q. Yang, and Z. Chen, "Document summarization using conditional random fields." in *IJCAI*, vol. 7, 2007, pp. 2862–2867.

[9] Y. Gong and X. Liu, "Generic text summarization using relevance measure and latent semantic analysis," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2001, pp. 19–25.

[10] Z. He, C. Chen, J. Bu, C. Wang, L. Zhang, D. Cai, and X. He, "Document summarization based on data reconstruction," in *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, 2012, pp. 620–626.

[11] M. Schmidt, "Least squares optimization with l1-norm regularization," *CS542B Project Report*, 2005.

[12] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[13] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM journal on scientific computing*, vol. 20, no. 1, pp. 33–61, 1998.

[14] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani *et al.*, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[15] T. T. Wu and K. Lange, "Coordinate descent algorithms for lasso penalized regression," *The Annals of Applied Statistics*, pp. 224–244, 2008.

[16] G. E. Hinton, "Learning distributed representations of concepts," in *Proceedings of the eighth annual conference of the cognitive science society*, vol. 1. Amherst, MA, 1986, p. 12.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

[18] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain, "Neural probabilistic language models," in *Innovations in Machine Learning*. Springer, 2006, pp. 137–186.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[20] Y. Liu, S.-h. Zhong, and W.-j. Li, "Query-oriented unsupervised multi-document summarization via deep learning," *Under review in Journal of Neural Networks (NN)*, 2008.

[21] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[22] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[23] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 2004, pp. 74–81.

[24] X. Li, S. Zhu, H. Xie, and Q. Li, "Document summarization via self-present sentence relevance model," in *Database Systems for Advanced Applications*. Springer, 2013, pp. 309–323.

[25] Z. Zhang, H. Li *et al.*, "Topicdsdr: combining topic decomposition and data reconstruction for summarization," in *Web-Age Information Management*. Springer, 2013, pp. 338–350.

[26] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.