Creating Reverse Dictionary of English Idiomatic Expressions by Mapping Word Embeddings to Singular Vectors

Xiaodong Liu, Rafal Rzepka, Kenji Araki
Faculty of Information Science and Technology
Hokkaido University
Sapporo, Japan
{xiaodongliu,rzepka,araki}@ist.hokudai.ac.jp

Abstract

This paper demonstrates a reverse dictionary that can return relevant idiomatic expressions based on queries (input descriptions) given by users. The implementation of the system can be achieved with a Vector Space Model (VSM). However, when it comes to VSM, although its performance on queries is high under the condition that pairs of common keywords are shared between a query and documents, the limitations such as lack of common words, and information loss after matrix factorization inevitably worsen the performance. To address these limitations, we employed a feed-forward neural network to map between rich literal semantics (word embeddings) and latent semantics (singular vectors) via general literal semantics (Bag-of-Words). A comparatively high performance on queries is achieved by this approach, which can address the limitations to some extent. Therefore, to take the advantages of both the VSM and the neural network, we sorted their outputs (cosine similarity between a query and vector representations of idiomatic expressions) in decreasing order, and significantly improved the performance on queries.

1 Introduction

According to the survey written by (Turney and Pantel, 2010), the VSM was developed for the SMART information retrieval system (Salton and others, 1971), which pioneered many of the concepts that are used in modern search engines (Schütze et al., 2008). One form of VSM is term-document matrix, which contains the documents representing phrases, sentences, or texts, and terms representing all the single words of those documents. The key idea of VSM is to represent all the documents in a term-document matrix as points in a space (distributional representations). The points that are close in the space can be considered as semantically similar, whereas the points that are far apart from each other are semantically distant. A user's query (pseudo-document) is also represented as a point in the same space as the documents. The documents are sorted in order of decreasing semantic similarity from the query, and then presented to the user.

The original elements in the vector representation of each document are Bag-of-Words. The frequencies of words in a document, to some extent, can determine how relevant the document is to a query. The Bag-of-Words hypothesis is the basis for applying the VSM to information retrieval (Salton et al., 1975). However, the original term-document matrix has some limitations when applied to information retrieval tasks. When the size of the matrix ($M \times N$) is enormous, this will consume a considerable memory footprint. Moreover, the documents used in our experiment are short texts such as phrases and sentences, which makes the matrix sparse. To address this limitation, matrix factorization was introduced, and one standard way to perform this mathematical operation is Singular Vector Decomposition (SVD) (Deerwester et al., 1990), which is engaged in the process of LSA (Latent Semantic Analysis) (Landauer and Dumais, 1997). SVD decomposes the matrix X ($M \times N$) into the product of three matrices U ($M \times K$) $\cdot \sum$ ($K \times K$) $\cdot V^T$ ($K \cdot N$), in which U is a left singular matrix, $\sum$ is a diagonal matrix of singular values, V^T is a right singular matrix, where $K \ll M$ or N. From the perspective of mathematics, this operation is a kind of dimension reduction. On the other hand, from the perspective of computational linguistics, we consider it as a latent semantics detection: U is composed of latent semantics of all words, and V^T is composed of la-

tent semantics of all documents. Proximity in the induced latent space has been shown to correlate with semantic similarity (Mihalcea et al., 2006). Within the right singular matrix, each document has k latent semantics, and then we just need to use some similarity mechanism like cosine similarity to judge which documents are semantically similar to a given query, and then display them to users. Some researchers argue that important information can be lost after matrix factorization (Ji and Eisenstein, 2013). This is one of the limitations that we want to address. We do not consider the latent semantics only, instead we bind it to literal semantics in order to realize a conceptual mapping mechanism. The details are presented in section 5.

Below is an example that shows another limitation of the VSM. Both descriptions are excerpted from two commercial dictionaries, which describe the meaning of the idiomatic expression: "have your cake and eat it too".

D1 "to get the benefits of two different situations or things when you should only get the benefit of one of them" (Collins Online Dictionary[1])

D2 "to have the advantages of something without its disadvantages" (Oxford Learner's Dictionary[2])

Since they describe the same idiomatic expression, they can be identified as paraphrase. However, common words are barely shared between them, and 'advantage' to 'benefit' and 'get' to 'have' do not share the same lemma. Therefore, it would be problematic for VSM to retrieve D2 when D1 is a query. We call this limitation lack of common words.

2 Reverse Dictionary

A reverse dictionary (Sierra, 2000), also known as an inverse dictionary, or search-by-concept dictionary (Calvo et al., 2016) is a system that returns words based on user descriptions or definitions (Zock and Bilac, 2004). For example, given input query "a large aggressive animal with wings and a long tail, that can breathe out fire", a reverse dictionary would return 'dragon' as primary candidate to users. A successful implementation of a reverse dictionary was proposed in 2016 (Hill et al., 2016) who used neural language embedding model to map dictionary definitions (phrases) to (lexical) representations of the words defined by those definitions, the performance of which could almost rival a commercial online reverse dictionary[3].

As defined above, a reverse dictionary of English idiomatic expressions is a system that returns idiomatic expressions to users, given a query of input description. As of writing, to the author's best knowledge, there is only one online commercial system[4] for looking up relevant idiomatic expressions given descriptions as queries. We also compared its performance on our test dataset. The implementation of the reverse dictionary of idiomatic expressions can be completed by VSM. However, to address the aforementioned limitations of VSM, our proposed method is presented in this paper.

3 Dataset

We have collected 1,404 common English idiomatic expressions and their corresponding descriptions (definitions) from three online resources[5]. In addition to these, we also added the definitions from Oxford Learner's Dictionary to the idiomatic expressions in our dataset. As a result, 2,330 descriptions have been collected. While some idiomatic expressions have only one description, others have two or more descriptions. For example, if the idiomatic expression w has descriptions $\{d_1 \ldots d_n\}$, then we include all pairs $(w, d_1) \ldots (w, d_n)$ as training examples. For the test dataset, we collected 70 input descriptions from Collins Online Dictionary as unseen data describing 70 idiomatic expressions randomly-chosen from our training dataset.

4 Baseline Method

LDA (Latent Dirichlet Allocation) (Blei et al., 2003) is a classic probabilistic model used to build up document-topics distribution and topic-terms distribution. In our experiment, only document-topics distribution was utilized. Since in our task of providing idiomatic expressions for input descriptions, texts are short phrases and sentences,

[1] https://www.collinsdictionary.com/

[2] https://www.oxfordlearnersdictionaries.com/definition/english/

[3] https://www.onelook.com

[4] http://www.idioms4you.com

[5] https://www.ef.com/wwen/english-resources/english-idioms/, https://7esl.com/english-idioms/, http://idiomsite.com/

measuring semantic similarity via topics distribution (100 topics in our baseline because it is same as the number of components of LSA set in the proposed method) would not be accurate. The vector representation (see Equation 1) of each idiomatic expression $\vec{I}$ is the averaged document-topics-distribution vectors $\vec{s}$ of all descriptions belonging to the idiomatic expression.

$$\vec{I} = \frac{\sum_{i=1}^{n} \vec{s_i}}{n} \quad (1)$$

5 Proposed Method

Our inspiration stems from connotations for a single word; for example, the word 'professional' has connotations of skill and excellence. Hence, if another single word like 'expert' also has similar connotations, we can consider them as semantically similar. Likewise, if two short texts like phrases or sentences have similar latent semantics, they can also be considered as semantically similar, and then they can be matched.

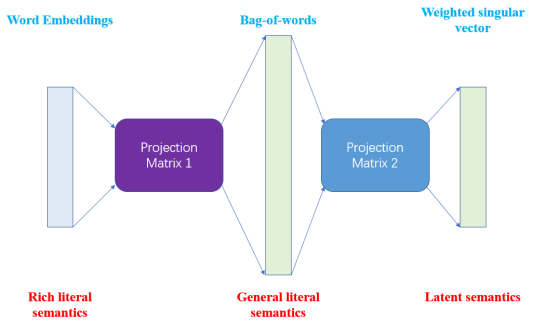

Figure 1: Our feedforward neural network

Figure 1 illustrates the mapping flow of our proposed, the original method described in Section 6. In this section, we focus on the introducing data structure, which is the prerequisite for the training process of the neural network.

The structure of the neural language model can address the limitation of lack of common words. Here is an example of our hypothesis.

- Rich literal semantics: *parents are euphoric about the news of his engagement.*
- General literal semantics: *his parents are excited about his engagement.*
- Latent Semantics: *positive sentiment; marriage*

Although WordNet (Miller, 1998) has well organized taxonomy of words (the word relationship like hyponym to hypernym can project the semantic specificity of words from 'rich' to 'general'), we decided to use GloVe (Pennington et al., 2014), which is considered as an improvement to word2vec model (Mikolov et al., 2013), and trained on billions of words of raw text. The reason that GloVe can outperform WordNet in our case is vector representation of words can be used as the layer in the flowchart of neural network, and elementwise addition of word embeddings; one paradigm is "king – man + woman = queen". Thus, the elementwise addition of the word embeddings of a short text is considered as rich literal semantics.

As for general literal semantics, in the raw term-document matrix, the 2,330 descriptions have 2,553 single words (with stop-words) and 2,436 single words (without stop-words), and each document (description in this work) is composed of Bag-of-Words. Therefore, the length of each document is 2,553 with stop-words, and 2,436 without stop-words. We use stop-word list provided by the Natural Language Toolkit (NLTK) (Loper and Bird, 2002).

After obtaining general literal semantics, we used "tf-idf reweighting with SVD" (100 components are set in SVD, which is equal to topics number of the baseline) to obtain the singular vectors of the decomposed singular matrix $(2,330 \times 100)$. With singular vectors, we can generate the vector representation of all idiomatic expressions. We call the vector representation "averaged singular vector" (namely averaged singular vectors generated by the equation of topics-distribution vectors in the baseline); from the perspective of computational linguistics, we consider all the elements in a vector representation as the latent semantics of the idiomatic expression.

As stated in many papers regarding classification of idioms, e.g. (Peng et al., 2018), idiomatic expressions exhibit the property of non-compositionality. Therefore, we did not use word embeddings or Bag-of-Words to represent idiomatic expressions. In addition, those two cannot represent latent semantics.

6 Implementation Details

Firstly, we trained the Projection Matrix 2 independently from the Projection Matrix 1, in or-

der to map between general literal semantics and latent semantics.

As we mentioned in Section 3, if an idiomatic expression w has n descriptions ($d_1 \ldots d_n$), then we include all pairs (w, d_1) ... (w, d_n) as training examples. For example, the input of Bag-of-Words d_1 is mapped to averaged-singular-vector(w); the same mapping is performed from d_2 to d_n. After all parameters of Projection Matrix 2 are settled after training, we train Projection Matrix 1.

While Projection Matrix 1 is being trained, the Bag-of-Words Layer is hidden layer with tanh, and the most important thing here is that all parameters in Projection Matrix 2 are not adjusted any more – it is fixed there just like pretrained GloVe. Identically as in the example given above, the input of elementwise-addition(d_1) is mapped to the averaged-singular-vector(w); the same mapping is performed from d_2 to d_n.

We trained two models: with and without stop-words to see if lack of them influence the results. In the case without stop-words, none of the three layers contains any stop-words.

During the test, given a description as an input query, a list of 80 idiomatic expressions is returned in decreasing order of cosine similarity between the output to the query and the averaged singular vectors of all idiomatic expressions.

To compare our model with a conventional VSM, we employed a raw term-document matrix for retrieving idiomatic expressions with and without stop-words; this approach was created to observe the performance change when only literal semantics is engaged. We also implemented LSA with tf-idf (reweighting the term-document matrix using tf-idf and decomposing the matrix using SVD afterwards) to investigate the performance when only latent semantics is utilized.

To combine the advantages of VSM and of our model, we integrated them and evaluated using 70 testing queries. The flowchart of testing a query is illustrated in Figure 2.

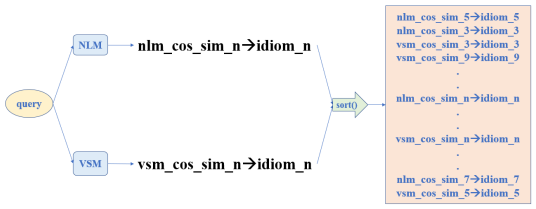

Figure 2: Integration of cosine similarity

The cosine similarities between a model output to a query and the vector representations of all idiomatic expressions are calculated at first, and then sorted into a list according to the decreasing order of cosine similarity.

Items	Settings
Activation Function	tanh
Input Length	200
Output Length	100
Hidden layer w stop-words	2,553
Hidden layer w/o stop-words	2,436
Loss function	squared error
Epochs (Projection Matrix 2)	200
Epochs (Projection Matrix 1)	2,000

Table 1: Neural network hyperparameters

7 Evaluation

Below we describe one observation on the output of our model with stop-words eliminated.

- Input Query: *"a situation of comfort or ease"*
- Idiomatic expression that ranks 4th in the output: *"a bed of roses"*
- The description in the training dataset: *"an easy or a pleasant situation"*

Above is only one observation of the output that ranks at 4th based on the query, which is the desired idiomatic expression that we want to look up. In addition to it, there are the observations that rank at 1st, 2nd or even 50th. Table 2 is not to evaluate but to show the overall performance of 70 queries of test dataset that each model returns.

In the table 2, "@1/10/30/50/80" means out of 70 outputs, what is the percentage that each desired idiomatic expression ranks over top 80, 50, 30, 10, or even exactly at 1.

8 Observation Analysis

Extrapolating from the result of our experiment, we can conclude that VSM performs well only when a query shares many common keywords with target documents; whereas our model can address the limitation of lack of common words. The integration of VSM and our proposed method has outperformed other methods as it takes the advantages of both VSM and our proposed method. Stop-words influence the result of VSM model to

Model	Model Description	@1	@10	@30	@50	@80
Baseline	LDA Topic Modeling	0%	1.49%	2.86%	4.29%	7.14%
LSA + tf-idf (w/o stop-words)	Latent Semantics	4.29%	25.71%	31.43%	38.57%	51.43%
LSA + tf-idf (with stop-words)		10%	35.71%	44.29%	52.86%	57.14%
VSM (w/o stop-words)	General Literal Semantics	37.14%	52.86%	58.57%	58.57%	58.57%
VSM (with stop-words)		32.85%	45.71%	50%	57.14%	58.57%
Proposed method (w/o stop-words)	Proposed Method	28.57%	51.42%	67.14%	72.86%	74.29%
Proposed method (with stop-words)		31.43%	51.42%	65.71%	70%	75.71%
Proposed method + VSM	Integration	**38.57%**	**62.86%**	**74.29%**	**80%**	**82.86%**
Idioms4u	Online Commercial	1.43%	20%	27.14%	31.43%	32.86%

Table 2: Comparison results

some extent. On the other hand, it holds almost no influence to our proposed method.

Below we describe three main observations. First, let us show an example that describes the advantage of VSM.

- Input query: *"a speech which is intended to encourage someone to make more effort or feel more confident"*
- The desired idiomatic expression: *"Pep talk"*
- Its descriptions in the training dataset: *"a short speech intended to encourage somebody to work harder"*, and *"an encouraging speech given to a person or group"*
- Baseline: not in the top 80 candidates
- VSM with stop-words involved: 1st
- VSM with stop-words eliminated: 1st
- LSA + tf-idf with stop-words involved: not in the top 80 candidates
- LSA + tf-idf with stop-words eliminated: not in the top 80 candidates
- Proposed method with stop-words involved: 37th
- Proposed method with stop-words eliminated: not in the top 80 candidates
- Proposed method + VSM: 5th
- Idioms4you: not in the top 80 candidates

"Speech", "intend", and "encourage" are the common keywords that can contribute to a high cosine similarity between the query and the descriptions. The mapping mechanism worked for one of our models (with stop-words) to some degree, but it did not outperform VSM.

The next example shows how our proposed method is able to address the lack of common words limitation.

- Input query: *"someone who does something or goes somewhere very early, especially very early in the morning."*
- The desired idiomatic expression: *"an early bird"*
- Its descriptions in the training dataset: *"A person who gets up early in the morning"*, *"A person who starts work earlier than others"*, and *"somebody who does something prior to the usual time"*
- Baseline: not in the top 80 candidates
- VSM with stop-words involved: not in the top 80 candidates
- VSM with stop-words eliminated: not in the top 80 candidates

- LSA + tf-idf with stop-words involved: 44th
- LSA + tf-idf with stop-words eliminated: not in the top 80 candidates
- Proposed method with stop-words involved: 23rd
- Proposed method with stop-words eliminated: 1st
- Proposed method + VSM: 23rd
- Idioms4you: not in the top 80 candidates

Our proposed method outperformed the VSM, even though one of them did not reach top 10. The performance is comparatively good, because the rich literal semantics was to some extent mapped to the general literal semantics. As for VSM, the only common word shared between the long query and the descriptions is 'early', so the performance was unsatisfactory.

Finally, the last example below demonstrates poor performance of all models.

- Input query: *"distressed or exasperated to the limit of one's endurance"*
- The desired idiomatic expression: *"at the end of one's tether"*
- Its descriptions in the training dataset: *"running out of endurance or patience"*
- Baseline: not in the top 80 candidates
- VSM with stop-words involved: not in the top 80 candidates
- VSM with stop-words eliminated: not in the top 80 candidates
- LSA + tf-idf with stop-words involved: 64th
- LSA + tf-idf with stop-words eliminated: not in the top 80 candidates
- Proposed method with stop-words involved: not in the top 80 candidates
- Proposed method with stop-words eliminated: not in the top 80 candidates
- Proposed method + VSM: not in the top 80 candidates
- Idioms4you: not in the top 80 candidates

In fact, none of the methods worked for this query. Our reasoning is that 'distress' and 'exasperate' are within the same cluster in the space of word embeddings, but in the description of the training dataset, there exists no general words like 'sad' that can be mapped from them. As for VSM, only one common word 'endurance' shared between the query and the description in a long phrase, so the cosine similarity between them was very low.

9 Conclusion and Future Work

In this paper, we presented a method for creating reverse dictionary of idiomatic expressions that can return relevant idiomatic expressions given input descriptions as queries. We used feedforward neural network to map between word2vec and singular vectors via Bag-of-Words, and extrapolating from the experimental results, this approach, to some extent, addresses the VSM's limitation of nonexistent common words. However, not all results of our proposed method achieved superior performance. Besides that, the size of dataset is small, which is not conducive to machine learning. Therefore, in future work, we are going to improve our method by adding idiomatic expressions from Wiktionary and enriching neural network architecture in order to further decreasing information loss after matrix factorization.

References

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Hiram Calvo, Oscar Méndez, and Marco A Moreno-Armendáriz. 2016. Integrated concept blending with vector space models. *Computer Speech & Language*, 40:79–96.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on*

Empirical Methods in Natural Language Processing, pages 891–896.

Thomas K Landauer and Susan T Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Aaai*, volume 6, pages 775–780.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Jing Peng, Anna Feldman, and Ekaterina Vylomova. 2018. Classifying idiomatic and literal expressions using topic models and intensity of emotions. *arXiv preprint arXiv:1802.09961*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Gerard Salton et al. 1971. The smart system—experiments in automatic document processing.

Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. Introduction to information retrieval. In *Proceedings of the international communication of association for computing machinery conference*, page 260.

Gerardo Sierra. 2000. The onomasiological dictionary: a gap in lexicography. In *Proceedings of the ninth Euralex international congress*, pages 223–235.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Michael Zock and Slaven Bilac. 2004. Word lookup on the basis of associations: from an idea to a roadmap. In *Proceedings of the Workshop on Enhancing and Using Electronic Dictionaries*, pages 29–35. Association for Computational Linguistics.