Semantic Word Embedding (SWE)

Reference Manual (Readme)

Open-source toolkits, knowledge collections and applications

May 25, 2015

Quan Liu
National Engineering Laboratory for Speech and Language Information Processing
University of Science and Technology of China
Website: http://home.ustc.edu.cn/~quanliu/

Contents

Abstract

SWE represents a general framework to incorporate semantic knowledge into the popular data-driven learning process of word embeddings to improve the quality of them. Under the SWE framework, semantic knowledge could be quantized as many **ordinal ranking inequalities** and the learning of word vectors is formulated as a **constrained optimization problem**. In detail, the data-derived objective function is optimized subject to all ordinal knowledge inequality constraints extracted from available knowledge resources such as Thesaurus, WordNet, knowledge graphs, etc. We have demonstrated that this constrained optimization problem can be efficiently solved by the stochastic gradient descent (SGD) algorithm, even for a large number of inequality constraints. Experimental results on four standard NLP tasks, including word similarity measure, sentence completion, name entity recognition, and the TOEFL synonym selection, have all demonstrated that the quality of learned word vectors can be significantly improved after semantic knowledge is incorporated as inequality constraints during the learning process of word embeddings.

Chapter 1 Semantic Word Embedding

1.1 The Skip-gram moel

The skip-gram model is a recently proposed learning framework *[10, 9]* to learn continuous word vectors from text corpora based on the aforementioned distributional hypothesis, where each word in vocabulary (size of V) is mapped to a continuous embedding space by looking up an embedding matrix $\mathbf{W}^{(1)}$. And $\mathbf{W}^{(1)}$ is learned by maximizing the prediction probability, calculated by another prediction matrix $\mathbf{W}^{(2)}$, of its neighboring words within a context window. Given a sequence of training data, denoted as $w_1, w_2, w_3, ..., w_T$ with T words, the skip-gram model aims to maximize the following objective function:

$$Q = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0} \log p(w_{t+j} \mid w_t) \tag{1}$$

where c is the size of context windows, w_t denotes the input central word and w_{t+j} for its neighbouring word. The skip-gram model computes the above conditional probability $p(w_{t+j} \mid w_t)$ using the following softmax function:

$$p(w_{t+j} \mid w_t) = \frac{\exp\left(\mathbf{w}_{t+j}^{(2)} \cdot \mathbf{w}_t^{(1)}\right)}{\sum_{k=1}^{V}\exp\left(\mathbf{w}_k^{(2)} \cdot \mathbf{w}_t^{(1)}\right)} \tag{2}$$

where $\mathbf{w}_t^{(1)}$ and $\mathbf{w}_k^{(2)}$ denotes row vectors in matrices $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$, corresponding to word w_t and w_k respectively.

1.2 SWE as Constrained Optimization

In the framework of SWE, each similarity ranking inequality involves a triplet, (i, j, k), of three words, w_i, w_j, w_k. Assuming the ordinal knowledge is represented by a large number of such inequalities, denoted as the inequality set S. For $\forall(i, j, k) \in S$, we have:

$$\begin{aligned} &\text{similarity}(w_i, w_j) > \text{similarity}(w_i, w_k) \\ \Leftrightarrow\ &\text{sim}\left(\mathbf{w}_i^{(1)}, \mathbf{w}_j^{(1)}\right) > \text{sim}\left(\mathbf{w}_i^{(1)}, \mathbf{w}_k^{(1)}\right) \end{aligned}$$

For notational simplicity, we denote $s_{ij} = \text{sim}\left(\mathbf{w}_i^{(1)}, \mathbf{w}_j^{(1)}\right)$ hereafter.

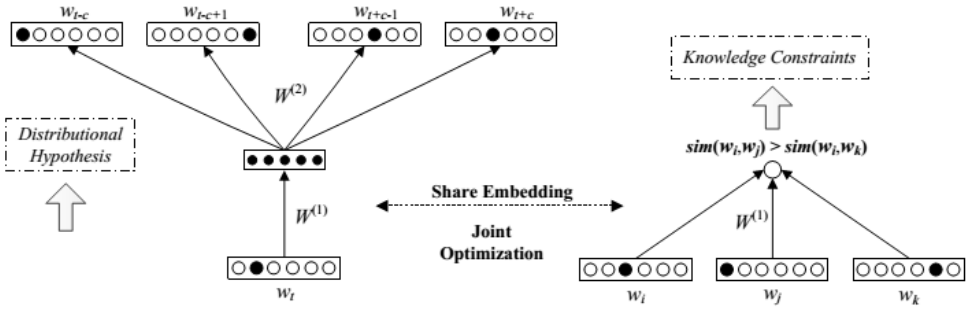


Figure 1: The proposed semantic word embedding (SWE) learning framework (The left part denotes the state-of-the-art skip-gram model; The right part represents the semantic constraints).

Next, we propose to use the following constrained optimization problem to learn semantic word embeddings (SWE):

$$\left\{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}\right\} = \arg\max_{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}} Q\left(\mathbf{W}^{(1)}, \mathbf{W}^{(2)}\right) \tag{3}$$

subject to

$$s_{ij} > s_{ik} \quad \forall (i, j, k) \in S \tag{4}$$

In this work, we formulate the above constrained optimization problem into an unconstrained one by casting all the constraints as a penalty term in the objective function. The penalty term can be expressed as follows:

$$D = \sum_{(i,j,k)\in S} f\left(i, j, k\right) \tag{5}$$

where the function $f(\cdot)$ is a normalization function. We set it to be a hinge loss function like $f(i, j, k) = h\left(s_{ik} - s_{ij}\right)$ where $h(x) = \max(\delta_0, x)$ with δ_0 denoting a parameter to control the decision margin. Finally, the proposed semantic word embedding (SWE) model aims to maximize the following combined objective function:

$$Q' = Q - \beta \cdot D \tag{6}$$

where β is a control parameter to balance the contribution of the penalty term in the optimization process.

In Figure 1, we show a diagram for the overall SWE learning framework to incorporate semantic knowledge into the basic skip-gram word embeddings. Comparing with the previous work in *[15]* and *[2]*, the proposed SWE framework is more general in terms of encoding the semantic knowledge for learning word embeddings. It is straightforward to show that the work in *[15, 17, 2]* can be viewed as some special cases under our SWE learning framework.

Chapter 2 Ordinal Semantic Constraints

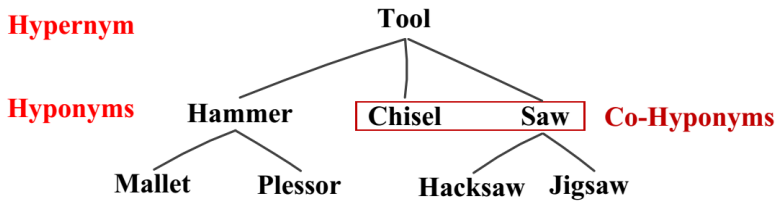
2.1 Representing Knowledge By Ranking

Many types of lexical semantic knowledge can be quantitatively represented by a large number of ranking inequalities such as:

$$\text{similarity}(w_i, w_j) > \text{similarity}(w_i, w_k) \quad (7)$$

where w_i, w_j and w_k denote any three words in vocabulary. For example, eq.(7) holds if w_j is a synonym of w_i and w_k is an antonym of w_i. In general, the similarity between a word and its synonymous word should be larger than the similarity between the word and its antonymous word. Moreover, a particular word should be more similar to the words belonging to the same semantic category as this word than other words belonging to a different category. Besides, eq.(7) holds if w_i and w_j have shorter distance in a semantic hierarchy than w_i and w_k do in the same hierarchy *[7, 5]*.

2.2 Three Common Sense Rules



✧ Figure 2: An example of hyponym and hypernym.

✧ ***Synonym Antonym Rule*:** Similarities between a word and its synonymous words are always larger than similarities between the word and its antonymous words. For example, the similarity between *foolish* and *stupid* is expected to be bigger than the similarity between *foolish* and *clever*, i.e., *similarity*(*foolish*, *stupid*) > *similarity*(*foolish*, *clever*).

✧ ***Semantic Category Rule*:** Similarities of words that belong to the same semantic category would be larger than similarities of words that belong to different categories. This rule refers to the idea of Fisher linear discriminant algorithm. A semantic category may be defined as a synset in WordNet, a hypernym in a semantic hierarchy, or an entity category in knowledge graphs. Figure 2 shows a simple example of the relationship between hyponyms and hypernyms. From there, it is reasonable to assume the following

similarity inequality: *similarity*(*Mallet*, *Plessor*) > *similarity*(*Mallet*, *Hacksaw*).

✧ ***Semantic Hierarchy Rule*:** Similarities between words that have shorter distances in a semantic hierarchy should be larger than similarities of words that have longer distances. In this work, the semantic hierarchy refers to the hypernym and hyponym structure in WordNet. From Figure 2, this rule may suggest several inequalities like: *similarity*(*Mallet*, *Hammer*) > *similarity*(*Mallet*, *Tool*).

Chapter 3 SWE Toolkit

3.1 Download and Installation

The SWE toolkit is now released under the Apache License, Version 2.0. This SWE toolkit is modified from the popular word2vec tool [link]. The toolkit includes the SWE training code, semantic knowledge constraints, application datasets, tools and scripts. For training SWE models and applying it for various NLP tasks, you need download it firstly and unpack it.

✧ Download SWE Toolkit.

✧ Find more information on GitHub page SWE-GitHub.

The SWE toolkit relies on the Intel math kernel library (MKL, download) for high efficiency. Once you have installed the MKL library, you can compile SWE as follows:

```
$ cd bin
$ [/bin] make
g++ SWE_Train.c -o SWE_Train -lm -pthread -O3 -march=native -Wall -funroll-loops
-Wno-unused-result -I/opt/intel/mkl/include -L/opt/intel/mkl/lib/intel64
-L/opt/intel/lib/intel64 -lmkl_intel_lp64 -lmkl_intel_thread -liomp5 -lmkl_core
g++ SWE_Test_SentComplete.c -o SWE_Test_SentComplete -lm -pthread -O3 -march=native -Wall
-funroll-loops -Wno-unused-result -I/opt/intel/mkl/include -L/opt/intel/mkl/lib/intel64
-L/opt/intel/lib/intel64 -lmkl_intel_lp64 -lmkl_intel_thread -liomp5 -lmkl_core
g++ SWE_Test_WordSim.cpp -o SWE_Test_WordSim -lm -pthread -O3 -march=native -Wall
-funroll-loops -Wno-unused-result -I/opt/intel/mkl/include -L/opt/intel/mkl/lib/intel64
-L/opt/intel/lib/intel64 -lmkl_intel_lp64 -lmkl_intel_thread -liomp5 -lmkl_core
g++ SWE_Test_SynSel.cpp -o SWE_Test_SynSel -lm -pthread -O3 -march=native -Wall
-funroll-loops -Wno-unused-result -I/opt/intel/mkl/include -L/opt/intel/mkl/lib/intel64
-L/opt/intel/lib/intel64 -lmkl_intel_lp64 -lmkl_intel_thread -liomp5 -lmkl_core
```

Figure 3. SWE compile procedures.

3.2 Main Tools and Functions

There are four major tools for supporting model training and applications in the SWE toolkit. Here are the detailed functions of each tool.

Table 1. SWE tool functions

Tool Name	Functions
SWE_Train	main tool, support SWE model as well as Skip-gram training.
SWE_Test_WordSim	tool for applying word embeddings for word similarity.
SWE_Test_SentComplete	tool for applying word embeddings for sentence completion
SWE_Test_SynSel	tool for applying word embeddings for synonym selection.

In addition to the application for the above-mentioned word similarity, sentence completion and synonym selection tasks, we apply the SWE word embeddings for the popular name entity recognition task according to the work "Word representations: A simple and general method for semi-supervised learning" (Turain, 2010). The corresponding tools, datasets and scripts for implementing this process can all be downloaded from CS.UIUC.

Chapter 4 SWE Applications

4.1 Word Similarity

4.1.1 Task description

Measuring word similarity is a traditional NLP task *[13]*. Here we compare several word embedding models on a popular word similarity task, namely WordSim-353 *[3]*, which contains 353 English word pairs along with human-assigned similarity scores, which measure the relatedness of each word pair on a scale from 0 (totally unrelated words) to 10 (very much related or identical words). The final similarity score for each pair is the average across 13 to 16 human judges. When evaluating word embeddings on this task, we measure the performance by calculating the Spearman rank correlation between the human judgments and the similarity scores computed based on the learned word embeddings.

4.1.2 SWE Demo Instruction

Here are the main steps for applying SWE word embeddings for word similarity task.

Demo: Semantic word embeddings for word similarity task

Step 1: prepare the training corpus and vocabulary

$ cd corpora

$ [/corpora] mkdir TEXT8

$ cd TEXT8

$ [/corpora/TEXT8] perl SWE_script_LearnVocab.pl

After getting all the words with occ frequency, cut them by 5 times.

training corpus: text8.txt

training vocabulary: text8.wordfreq.cut5

Step 2: prepare the semantic constraint collections

$ cd semantics

$ [/semantics] mkdir TEXT8

$ cd TEXT8

$ [/semantics/TEXT8] perl SWE_script_IneqFilterByVocab.pl

You could find three semantic constraint sets:

(1) Synon-Anton: SemWE.EN.KnowDB.SA1.inTEXT8

(2) Hyper-Hypon: SemWE.EN.KnowDB.HH1.inTEXT8

(3) Both: SemWE.EN.KnowDB.COM1.inTEXT8

Step 3: setting word embedding parameters and training

$ cd task1_wordsim

$ [/task1_wordsim] vim SWE_Train_TEXT8.pl

Key parameter: @inter_param = (0, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5);

After setting the correct paths and parameters, you could the train SWE models.

$ [/task1_wordsim] perl SWE_Train_TEXT8.pl

Step 4: model evaluation

$ [/task1_wordsim] perl SWE_Test_WordSim.pl

This script would invoke the tool SWE_Test_WordSim in the bin directory.

Final evaluation result file: EmbedVector_TEXT8.WordSim353.result

For a quick demo run, you may get similar evaluation results as follows.

Table 2. Demo: Spearman results on the WordSim-353 Task.

	Word Embeddings	**Result**
TEXT8	Skip-gram	0.4666
	SWE + Synon-Anton	0.5266
	SWE + Hyper-Hypon	0.5083
	SWE + Both	0.5449

From the results in Table 2, we can see that the proposed SWE word embedding model can achieve consistent improvements over the baseline skip-gram model. In order to obtain state-of-the-art performances on this task, please use larger training corpora, such as the popular English Wikipedia, Google 1 Billion LM training corpus, etc.

4.2 Sentence completion

4.2.1 Task description

The Microsoft sentence completion challenge has recently been introduced as a standard benchmark task for language modeling and other NLP techniques *[16]*. This task consists of 1040 sentences, each of which misses one word. The goal is to select a word that is the most coherent with the rest of the sentence, from a list of five candidates. Many NLP techniques have already been reported on this task, including N-gram model and LSA-based model proposed in *[16]*, log-bilinear model *[12]*, recurrent neural networks (RNN) *[11]*, the skip-gram model *[9]*, a combination of the skip-gram and RNN model, and a knowledge enhanced word embedding model proposed by Bian et. al. *[1]*.

4.2.2 SWE Demo Instruction

To run a quick demo of applying SWE model for sentence completion, we follow the the same procedure as in *[9]*. The training corpus is obtained from the state-of-the-art work *[16]*. Here are the major steps for running this demo.

Demo: Semantic word embeddings for sentence completion task

Step 1 and Step 2 are the same to the aforementioned procedures in WordSim-353 task.

(a) Training corpus: /corpora/Holmes/Holmes_Training_Data.txt

(b) Training vocabulary: /corpora/Holmes/Holmes_Training_Data.txt.wordfreq.cut5

(1) Semantic constraint Synon-Anton: SemWE.EN.KnowDB.SA1.inHolmes

(2) Semantic constraint Hyper-Hypon: SemWE.EN.KnowDB.HH1.inHolmes

(3) Semantic constraint Both: SemWE.EN.KnowDB.COM1.inHolmes

Step 3: setting word embedding parameters and training

$ cd task2_microsoft

$ [/task2_microsoft] vim SWE_Train_Holmes.pl

Key parameter: @inter_param = (0, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5);

After setting the correct paths and parameters, you could the train SWE models.

$ [/task2_microsoft] perl SWE_Train_Holmes.pl

Step 4: model evaluation

$ [/task1_wordsim] perl SWE_Test_SentCompletion.pl

This script would invoke the tool SWE_Test_SentComplete in the bin directory.

Final evaluation result file: EmbedVector_Holmes.Holmes.result

The quick demo results are:

	System	**Acc**
Demo run	Skip-gram	31.8
	SWE + Synon-Anton	40.3
	SWE + Hyper-Hypon	40.0
	SWE + Both	41.4

Table 3. Demo: Results on Sentence Completion Task.

In Table 3, we have shown the sentence completion accuracy on this task for various word embedding models. We can see that the proposed SWE model has achieved considerable improvements over the baseline skip-gram model.

4.3 Name entity recognition

4.3.1 Task description

Applying SWE word embeddings for the standard CoNLL03 name entity recognition (NER) task *[14]*. The CoNLL03 NER dataset is drawn from the Reuters newswire. The training set contains 204K words (14K sentences, 946 documents), the test set contains 46K words (3.5K sentences, 231 documents), and the development set contains 51K words (3.3K sentences, 216 documents).

4.3.2 SWE Demo Instruction

To conduct experiments on this task, we use the same configurations as in *[14]*, including the used training algorithm, the baseline discrete features and so on. The training corpus contains one year of Reuters English newswire from RCV1, from August 1996 to August 1997, having about 810,000 news stories *[8]*.

- The training corpus RCV1 is available at NIST-RCV1.
- Configuration tools are available at CS.UIUC. You can also download them directly at NerACL2010_Experiments.zip.

Demo: Semantic word embeddings for name entity recognition task

Step 1 and Step 2 are the same to the aforementioned procedures in WordSim-353 task.

(a) Training corpus: /corpora/Reuters/reuters.rcv1.corpus

(b) Training vocabulary: /corpora/Reuters/reuters.rcv1.corpus.wordfreq.cut5

(1) Semantic constraint Synon-Anton: SemWE.EN.KnowDB.SA1.inReuters

(2) Semantic constraint Hyper-Hypon: SemWE.EN.KnowDB.HH1.inReuters

(3) Semantic constraint Both: SemWE.EN.KnowDB.COM1.inReuters

Step 3: setting word embedding parameters and training

$ cd task3_conll03ner

$ [/task3_conll03ner] vim SWE_Train_ReutersNER.pl

Key parameter: @inter_param = (0, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5);

After setting the correct paths and parameters, you could the train SWE models.

$ [/task3_conll03ner] perl SWE_Train_ReutersNER.pl

Step 4: applying SWE models for NER task, configuration.

Before running this script, you need to download the "NerACL2010_Experiments.zip" package from CS.UIUC. After that, unpacking it at the "task3_conll03ner" directory. Then copying subdir Cw50Dim0.3 as to NerACL2010_Experiments/bin.

Not easy? Try to download the package here (NerACL2010_Experiments.zip) and unpack it.

$ [/task3_conll03ner] perl SWE_Test_ConfigForNER.pl

Step 5: manage all the SWE experimental results

$ [/task3_conll03ner] perl SWE_script_ManageResult.pl

For example, the quick demo results are:

	System	Dev	Test	MUC7
Others	C&W	92.3	87.9	75.7
Demo run	Skip-gram	92.6	88.0	77.1
	SWE + Synon-Anton	92.5	88.5	77.3
	SWE + Hyper-Hypon	92.5	88.4	77.4
	SWE + Both	92.5	88.2	77.8

Table 4. Demo: F1 scores on the CoNLL03 NER task.

From the quick demo experimental results shown in Table 4, we find the SWE model can achieve better improvements on the MUC7 task than on the test set. To make further investigations, it is necessary to consider more semantic constraints for improving the SWE performances on NER task.

4.4 Synonym selection

4.4.1 Task description

The goal of the popular synonym selection task is to select, from a list of candidate words, the semantically closest word for each given target word. The dataset we use for this task is the standard TOEFL dataset *[6]*, which contains 80 questions. Each question consists of a target word along with 4 candidate lexical substitutes for selection. The evaluation criterion on this task is the synonym selection accuracy which indicates how many synonyms are correctly selected for all those questions.

4.4.2 SWE Demo Instruction

The major steps for applying SWE word embeddings for synonym selection task are exactly the same as the steps proposed in the word similarity task.

Demo: Semantic word embeddings for synonym selection task

For simplicity, here we use the same model trained in word similarity application.

Step for model evaluation

```
$ cd task4_synselect
$ [/task4_synselect] perl SWE_Test_SynSel.pl ../task1_wordsim/EmbedVector_TEXT8
```

This script would invoke the tool SWE_Test_SynSel in the bin directory.

Final evaluation result file: TOEFL80.result

Corpus	Model	Accuracy (%)
TEXT8	Skip-gram	50.00
	SWE + Synon-Anton	55.00
	SWE + Hyper-Hypon	53.75
	SWE + Both	55.00

Table 5. Demo: The TOEFL synonym selection task.

In Table 5, we have shown the experimental results for different word embedding models. From the experimental results in Table 5, we can see that the SWE model achieves consistent improvements over the Skip-gram model on the synonym selection task. Again, we suggest to use larger training corpora for conducting more experiments under the SWE framework.

5. Final Remarks

Word embedding models with good semantic representations are quite invaluable to many natural language processing tasks. However, the current data-driven methods that learn word vectors from corpora based on the distributional hypothesis tend to suffer from some major limitations. The proposed SWE is a general and flexible framework for incorporating various types of semantic knowledge into the popular data-driven learning procedure. **The main contribution of SWE is to represent semantic knowledge as a number of ordinal similarity inequalities as well as to formulate the entire learning process as a constrained optimization problem**.

References

- [1] J. Bian, B. Gao and T. Liu (2014) Knowledge-powered deep learning for word embedding. Machine Learning and Knowledge Discovery in Databases, pp. 132–148. Cited by: 4.2.1.
- [2] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy and N. A. Smith (2014) Retrofitting word vectors to semantic lexicons. Cited by: 1.2.
- [3] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman and E. Ruppin (2001) Placing search in context: the concept revisited. pp. 406–414. Cited by: 4.1.1.
- [4] R. A. Fisher (1936) The use of multiple measurements in taxonomic problems. Annals of eugenics 7 (2), pp. 179–188. Cited by: 2.2.
- [5] D. Jurafsky and J. H. Martin (2000) Speech & language processing. Pearson Education India. Cited by: 2.1.
- [6] T. K. Landauer and S. 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), pp. 211. Cited by: 4.4.1.
- [7] C. Leacock and M. Chodorow (1998) Combining local context and wordnet similarity for word sense identification. WordNet: An electronic lexical database 49 (2), pp. 265–283. Cited by: 2.1.
- [8] D. D. Lewis, Y. Yang, T. G. Rose and F. Li (2004) RCV1: a new benchmark collection for text categorization research. Journal of Machine Learning Research 5, pp. 361–397. Cited by: 4.3.2.
- [9] T. Mikolov, K. Chen, G. Corrado and J. Dean (2013) Efficient estimation of word representations in vector space. Cited by: 1.1, 4.2.1, 4.2.2.
- [10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean (2013) Distributed representations of words and phrases and their compositionality. pp. 3111–3119. Cited by: 1.1.
- [11] T. Mikolov (2012) Statistical language models based on neural networks. Ph.D. Thesis, Ph. D. thesis, Brno University of Technology. Cited by: 4.2.1.
- [12] A. Mnih and Y. W. Teh (2012) A fast and simple algorithm for training neural probabilistic language models. Cited by: 4.2.1.

- [13] H. Rubenstein and J. B. Goodenough (1965) Contextual correlates of synonymy. Communications of the ACM 8 (10), pp. 627–633. Cited by: 4.1.1.
- [14] J. Turian, L. Ratinov and Y. Bengio (2010) Word representations: a simple and general method for semi-supervised learning. pp. 384–394. Cited by: 4.3.1, 4.3.2.
- [15] C. Xu, Y. Bai, J. Bian, B. Gao, G. Wang, X. Liu and T. Liu (2014) RC-net: a general framework for incorporating knowledge into word representations. pp. 1219–1228. Cited by: 1.2.
- [16] G. Zweig and C. J. Burges (2011) The microsoft research sentence completion challenge. Technical report Technical Report MSR-TR-2011-129, Microsoft. Cited by: 4.2.1, 4.2.2.
- [17] G. Zweig (2014) Explicit representation of antonymy in language modelling. Technical report Technical Report MSR-TR-2014-52, Microsoft. Cited by: 1.2.