Refining Word Embeddings for Sentiment Analysis

Liang-Chih Yu, Jin Wang, K. Robert Lai and Xuejie Zhang 2017

Louis (Yiqing) Luo

July 13th, 2018

Current Word Vector Models

- Current models do not distinguish words with contrasting sentiments
  - Table below come from GoogleNews Negative 300.
  - Words with opposite sentiment polarities may have similar vector representations.
    - Building on such ambiguous vectors will affect sentiment classification performance.

Word 1	**Word 2**	**Similarity**
happy	sad	0.5354614
love	hate	0.6003957
smart	stupid	0.47047195

Proposed Strategy

- Strategy: Adjust the pre-trained vector of each affective word in a given sentiment lexicon such that it can be closer to a set of both semantically and sentimentally similar nearest neighbors and further away from sentimentally dissimilar neighbors.
  - Applicable to all pre-trained word representation models as a post-processing step.

Model in Detail

How does the model work?

1. Calculate the semantic similarity between target words and other words in lexicon based on cosine distance in the pretrained vector space

Model in Detail

How does the model work?

1. Calculate the semantic similarity between target words and other words in lexicon based on cosine distance in the pretrained vector space
2. Select top "k" similar words as neighbours

Model in Detail

How does the model work?

1. Calculate the semantic similarity between target words and other words in lexicon based on cosine distance in the pretrained vector space
2. Select top "k" similar words as neighbours
3. Re-rank using a newly assigned sentiment score from sentiment lexicon.
   - Ranked using an extend version of Affective Norms of English Words (E-ANEW).
   - Each word ϵ [1,9] in R^3

Model in Detail

How does the model work?

1. Calculate the semantic similarity between target words and other words in lexicon based on cosine distance in the pretrained vector space
2. Select top "k" similar words as neighbours
3. Re-rank using a newly assigned sentiment score from sentiment lexicon.
   - Ranked using an extend version of Affective Norms of English Words (E-ANEW).
   - Each word ϵ [1,9] in R^3
4. Refined to be closer to its semantically and sentimentally similar nearest neighbors and further away from sentimentally dissimilar neighbors.

Example of Steps 1 - 3

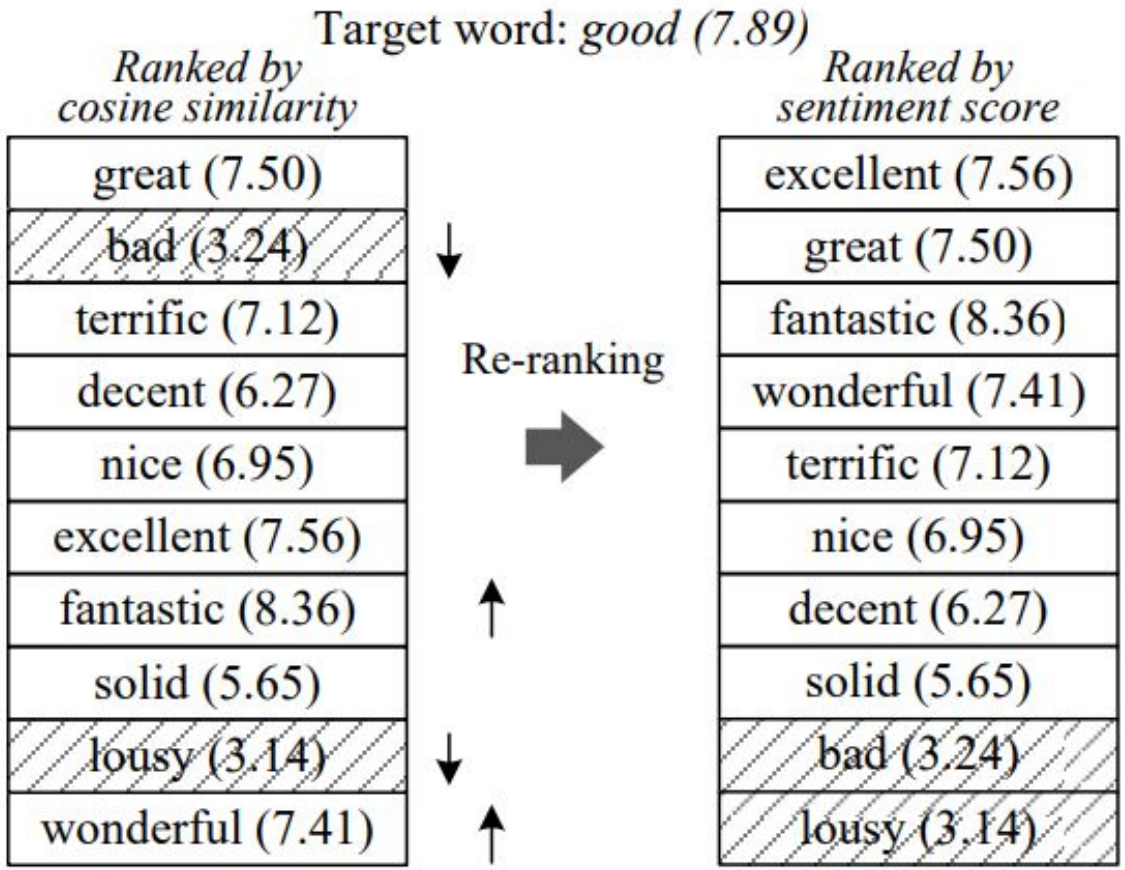

Figure: Example of nearest neighbor ranking.

Refinement Model

- Refinement Criteria:
  1. closer to its sentimentally similar neighbors,
  2. further away from its dissimilar neighbors, and
  3. not too far away from the original vector.

Refinement Model

- Refinement Criteria:
  1. closer to its sentimentally similar neighbors,
  2. further away from its dissimilar neighbors, and
  3. not too far away from the original vector.

$$\arg\min \Phi(V) = \arg\min \sum_{i=1}^{n} \left[\alpha dist(v_i^{t+1}, v_i^t) + \beta \sum_{j=1}^{k} w_{ij} dist(v_i^{t+1}, v_j^t) \right]$$

- $V = \{v_1, ...v_n\}$ - set of pretrained vectors corresponding to the word affective words in lexicon
- $\text{dist}((v_i, v_j)) = \|\mathbf{v_i} - \mathbf{v_j}\|_2$
- $w_{ij} = \frac{1}{\text{rank of word j in sentence}}$
- $\alpha, \beta =$ ratio used to control how much movement in each iteration of refinement

Refinement Model

- Through the iterative procedure, the vector representation of each target word will be iteratively updated until the change of the location of the target words vector is converged
- For word i:

$$v_i^{t+1} = \frac{\gamma v_i^t + \beta \sum_{j=1}^{k} w_{ij} v_j^t}{\gamma + \beta \sum_{j=1}^{k} w_{ij}}$$

Refinement Example

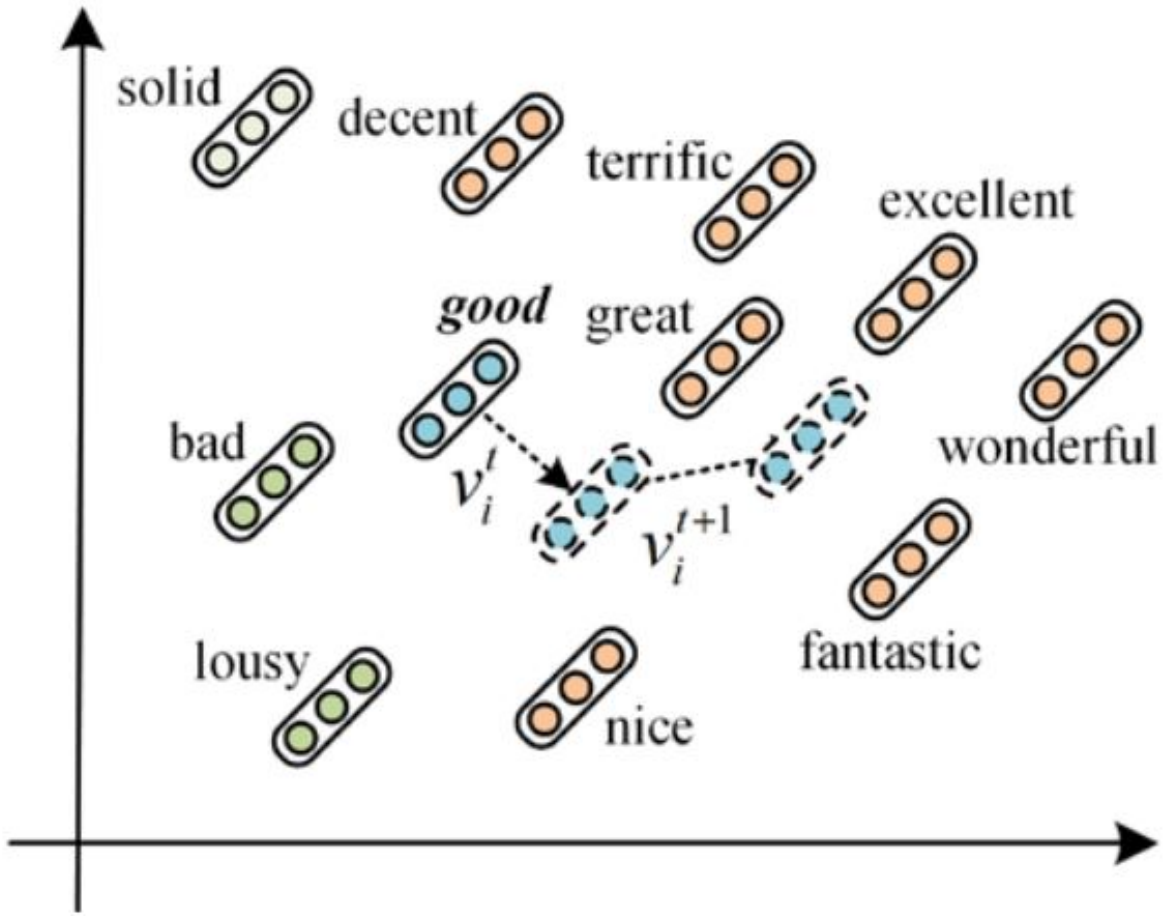

Figure 2: Conceptual diagram of word vector refinement.

Evaluation

- Evaluated by examining whether our refined embeddings can improve conventional word embeddings and outperform previously proposed sentiment embeddings
  - Dataset: Stanford Sentiment Treebank

Improvement in Results

Method	Fine-grained	Binary
DAN		
- Word2vec	46.2	84.5
- GloVe	46.9	85.7
- Re(Word2vec)	48.1	87.0
- Re(GloVe)	**48.3**	**87.3**
- HyRank	47.2	86.6
CNN		
- Word2vec	48.0	87.2
- GloVe	46.4	85.7
- Re(Word2vec)	**48.8**	**87.9**
- Re(GloVe)	47.7	87.5
- HyRank	47.3	87.6
Bi-LSTM		
- Word2vec	48.8	86.3
- GloVe	49.1	87.5
- Re(Word2vec)	49.6	88.2
- Re(GloVe)	**49.7**	**88.6**
- HyRank	49.0	87.3
Tree-LSTM		
- Word2vec	48.8	86.7
- GloVe	51.8	89.1
- Re(Word2vec)	50.1	88.3
- Re(GloVe)	**54.0**	**90.3**
- HyRank	49.2	88.2

Table 1: Accuracy of different classifiers with different word embeddings for binary and fine-grained classification.

Word Embeddings	Noise@10 (%)
Word2vec	24.3
GloVe	24.0
HyRank	18.5
Re(Word2vec)	14.4
Re(GloVe)	13.8

Table 2: Average percentages of noisy words in the top 10 nearest neighbors for different word embeddings.

(b)