Word Embeddings using Multiple Word Prototypes

Anurendra Kumar, Nishant Rai

Indian Institute of Technology Kanpur

Introduction

Learn multiple embeddings taking polysemy into account.
Rising interest in vector space word embeddings and their use, given recent methods for their fast estimation at very large scale.

Drawback : Almost all recent works assume a single representation for each word type, completely ignoring polysemy which leads to errors.

Examples :
- I can hear 'bass' sounds, They like grilled 'bass'.
- What does a 'bat' eat, Ram hit him with a 'bat'.

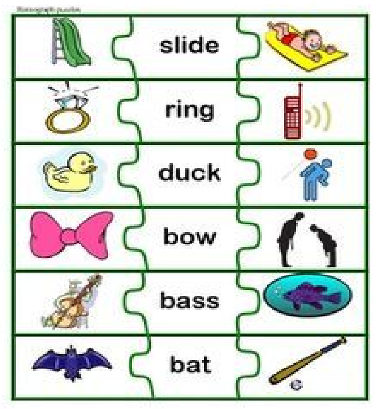

Figure 1 :Multi Sense Words [6]

Previous Work

- Mooney et al. [1] introduce a method for constructing multiple vector representations of words.
- Huang et al. [2] extend this approach incorporating global document context to learn multiple dense, low-dimensional embeddings by using recursive neural networks.
- Both the methods perform word sense discrimination as a preprocessing step by clustering contexts for each word type, making training more expensive.
- Improvements are suggested in the methods proposed by Neelakantan et al. [3], in which multiple word senses and global representations are computed simultaneously. This is one of the first papers which explore Non Parametric Word Embeddings.

Approach

1. Input: Single sense word embeddings OR Construction of word embeddings [3]
2. Identify top M words for which we compute multiple senses (Generally by frequency)
3. Construct context vectors; Estimate the optimal number of senses (clusters) OR Use the given parameter.
4. Perform clustering using the estimate.
5. Use cluster centres as sense vectors [1]

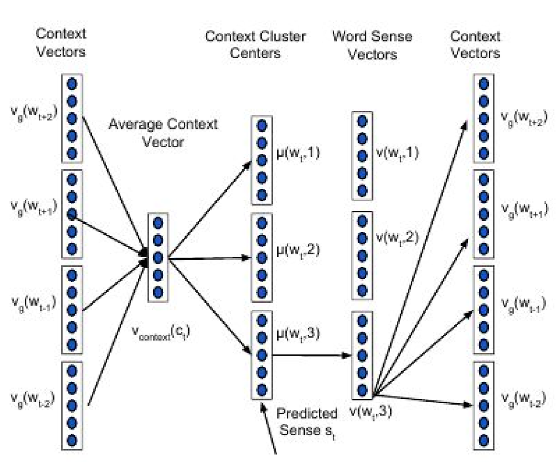

Figure 2 : MSSG Model [3]

Further Improvements,

A. Use the estimated sense vectors to assign senses to all occurrences of the words; Retrain using skip gram model.
B. Directly use cluster senses as vectors. The global word vector becomes the average of the sense vectors. (The final vector space remains the same) [1]

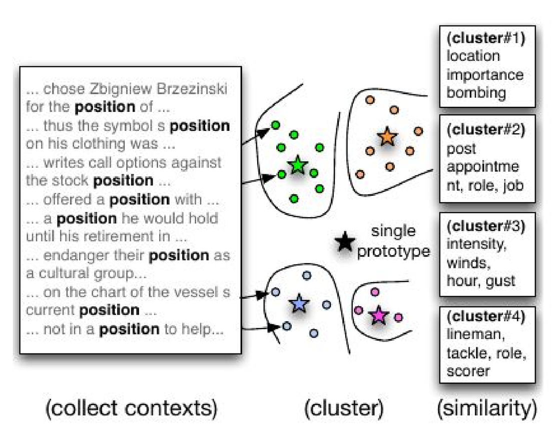

Figure 3 : Single Embeddings [1]

Methodology

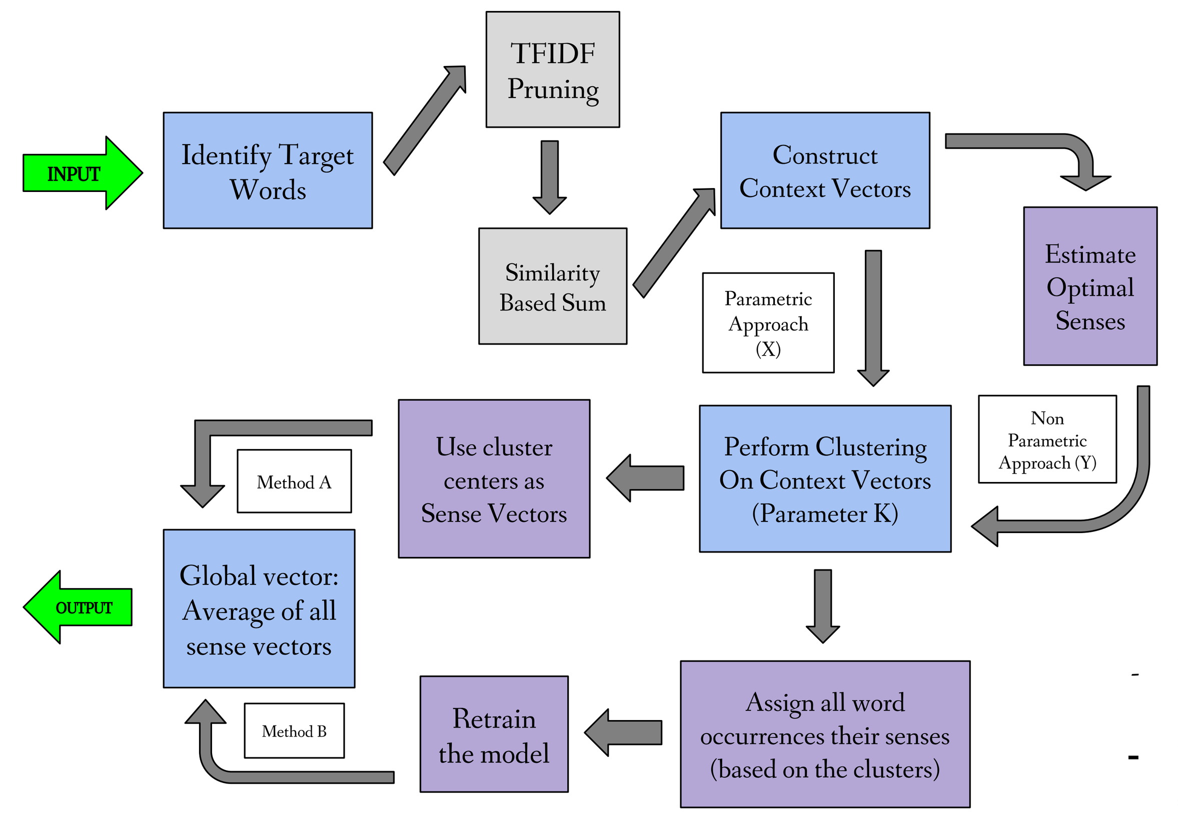

Figure 4 : Flowchart of the multiple Approaches followed

$$J(\theta) = \sum_{(w_t,c_t)\in D^+} \sum_{c\in c_t} \log P(D=1|v(w_t),v(c)) + \sum_{(w_t,c'_t)\in D^-} \sum_{c'\in c'_t} \log P(D=0|v(w_t),v(c'))$$

Transforming Objective Function to adapt to Multiple Senses

$$J(\theta) = \sum_{(w_t,c_t)\in D^+} \sum_{c\in c_t} \log P(D=1|v_s(w_t,s_t),v_g(c)) + \sum_{(w_t,c'_t)\in D^-} \sum_{c'\in c'_t} \log P(D=0|v_s(w_t,s_t),v_g(c'))$$

- The modified cost function used for estimating optimal number of senses. Estimation aimed at finding the best cost.
- TFIDF Pruning: Consider only influential words (based on TFIDF) i.e. remove words with low TFIDF.
- Similarity Based Sum: During context vector computation, consider words similar (e.g. Cosine Similarity) with the current word.

Semantic Similarity Metrics

$$\text{globalSim}(w,w') = d\left(v_g(w), v_g(w')\right) \quad \text{localSim}(w,w') = d\left(v_s(w,k), v_s(w',k')\right)$$

$$\text{avgSim}(w,w') = \frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K} d\left(v_s(w,i), v_s(w',j)\right)$$

AvgSim also an isolated word similarity metric (Since it does NOT take context into account)

AvgSimC considers context

$$\text{avgSimC}(w,w') = \sum_{j=1}^{K}\sum_{i=1}^{K} P(w,c,i)P(w',c',j) \times d\left(v_s(w,i), v_s(w',j)\right)$$

Results

Word	Nearest Neighbors (Model Y B - 50d (Neel))
plant #0	factory, refinery, facility, company, smelter, dearborn, hydro, furnace, brewing, manufacturing, laboratory
plant #1	animal, seedling, seed, fungus, algae, legume, edible, nitrogen-fixing, maize, vegetable, insect, invertebrate
space #0	core, vehicle, surface, craft, frame, plane, atmosphere, storage, orbit, energy, memory, matrix, deck, vision
space #1	dimension, surface, rotation, field, core, transformation, projection, mirror, grid, map, sphere, gravity, mass
space #2	nasa, NASAs, Raffaello, research, shuttle, Multi-Purpose, rocket, installation, mission, suborbital, satellite, ESA's
apple #0	microsoft, macintosh, acorn, intel, toaster, ibm, apple's, oracle, sony, motorola, OS, corel, netscape, ATI, AMD
apple #1	ice, sour, cabbage, crispy, sparkling, nut, cream, tomato, guava, chocolate, seaweed, strawberry, pie, hostess, oatmeal

Model	Time (In hours)
Huang et al	168
MSSG - 50d	1
NP-MSSG - 50d	1.83
Skip-gram - 50d	0.33
MSSG - 300d	6
NP-MSSG - 300d	5
Skip-gram - 300d	1.5
Model X A - 50d	6
Model X B - 50d	11.36
Model Y A - 50d	8
Model Y B - 50d	13.37

Code Implemented in Python
(Main process slowdown in File I/O)
(Expect upto 3-4 times faster run time in C++)
Sense Distinction not shown in the Nearest Neighbors result for the sake of clarity

Evaluation

Model	avgSim	globalSim	avgSimC	localSim
Huang et al - 50d	62.8*	58.6*	65.7*	26.1*
Model X A - 50d (Huang, K = 3)	55.0	45.0	55.0	38.6
Model Y A - 50d (Huang)	52.3	45.0	53.2	38.1
Model Y B - 50d (Huang)	**63.3**	**62.8**	**63.9**	**57.6**
Model Y B - 50d (Huang, With TFIDF Pruning)	**63.3**	**62.8**	63.3	55.9
Model X B - 50d (Huang, K = 3)	63.0	62.4	63.5	51.4
Neel et al: MSSG - 50d	**64.2**	62.1	**66.9**	49.2
Neel et al: NP-MSSG - 50d	64.0	62.3	66.1	50.3
Model X A - 50d (Neel, K = 3)	55.0	62.3	57.1	42.2
Model Y A - 50d (Neel)	48.3	62.3	38.1	32.8
Model Y B - 50d (Neel)	63.1	**62.4**	63.3	51.1
Model Y B - 50d (Neel, With TFIDF Pruning)	63.1	**62.4**	63.2	**61.0**
Model X B - 50d (Neel, K = 3)	63.0	61.8	64.3	50.6

Values represent Spearman correlation after multiplying by 100

* not able to recreate the results

Conclusions

- Our model achieves results which are comparable to state of the art results. Also able to outperform the current state of the art at LocSim Metric.
- Important Implication: Able to correctly identify senses, which has not been successfully performed previously, shown in low LocSim scores of other models.
- The accuracies of Method A (which gives output in the original vector space) depends largely on the initial seed vectors provided (It performs well for *Huang* but not for *Neel*)

* *Huang, Neel*: Word Vectors in [2], [3]

Future Work

- Improved Non Parametric clustering: Employ a clustering model with infinite capacity, e.g. the Dirichlet Process Mixture Model [5]. Allow more polysemous words to adopt more representations. Tackles the issue of varying word senses.
- Cluster similarity metrics: Other similarity metrics over mixture models, e.g. KL-divergence, with possibly better correlation with human similarity judgements.
- Better representation for contexts: Computing context vectors which represent word contexts in a better way would help improve the quality of the clusters.
- Joint model for clustering: Current method independently clusters the contexts of each word, so the senses discovered for w cannot influence the senses discovered for w'. Sharing such information could yield better results.

References

1. Reisinger, Joseph, and Raymond J. Mooney. "Multi-prototype vector-space models of word meaning." Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2010.
2. Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, pages 873–882. Association for Computational Linguistics, 2012.
3. Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. arXiv preprint arXiv:1504.06654, 2015.
4. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.
5. Rasmussen, Carl Edward. "The infinite Gaussian mixture model." NIPS. Vol. 12. 1999, pages 554--560.
6. Image taken from : https://www.pinterest.com/coupondarling/teaching-ideas/