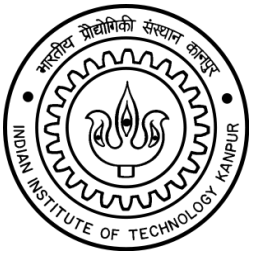
Word Embeddings with Multiple Word Prototypes

Nishant Rai[1] and Anurendra Kumar[2]

[1] Department of Computer Science and engineering
[2] Department of Electrical Engineering

Abstract

The project deals with the problem of construction of multiple sense embeddings for different words. We develop several models to tackle the problem consisting of online clustering Methods, methods involving parameter estimation and also look at methods related to word word co-occurrence matrices. The training data used is the April 2010 snapshot of the Wikipedia corpus [18]. Our model uses the popular Word2vec tool [1] to compute single word embeddings. We compare the performance of our approaches with current state of the art models [9] [14]and find that our model is comparable to the state of the art models and even outperforms them in some cases. Our model is extremely efficient and takes less than 6 hours for complete training and computation of senses. We also discuss the possibility of our model giving better (semantically coherent) senses than present models. The main task used for comparing the models is the SCWS task [9]. The comparisons have been done using human judgment of semantic similarity.



Under the guidance of Dr. Amitabha Mukherjee
Indian Institute of Technology, Kanpur

Contents

1 Introduction

The ability to accurately represent word vectors to capture syntactic and semantic similarity is central to Natural language processing. Thus, there is rising interest in vector space word embeddings and their use especially given recent methods for their fast estimation at very large scale [12] [16]. However almost all recent works assume a single representation for each word type, completely ignoring polysemy which eventually leads to errors.

Problems caused by ignoring polysemy can be seen in the following sentences,

- "I can hear bass sounds"
- "They like grilled bass"

Here, the word bass represents two different meanings: tones and a type of fish, respectively. Making this distinction is not possible unless we account for polysemy in our models.

2 Motivation

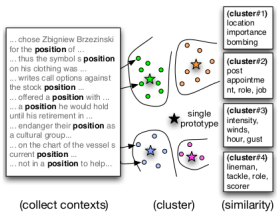
Inspiration to compute multiple embeddings comes from the following observation, a word with two distinct senses A and B will have neighbors belonging to two distinct clusters. Notice that even though the distance $d(word, A)$ and $d(word, B)$ are small, $d(A, B)$ might be large in case A and B are completely different senses. This can be termed as violation of triangle inequality [17]. Many approaches exploit the property to compute better representations of word types.

Notice that word embeddings play a crucial role in construction of multiple embeddings, thus we should also aim at improving the word embeddings. In nearly all approaches, word vectors are built from local contexts. But in work done by Huang et al. [9], it is shown that including global context helps improve the quality of the embeddings.

In our project, try to compute multiple senses of a given word, taking inspiration from work done by Mooney et al [17], Huang et al [9], Neelankantan et al [14].

3 Word Vector Representation

Word vectors are mapping from words to some high-dimensional space where each word is assigned a unique vector such that similar words are near and dissimilar words are far apart using some distance metric generally cosine [6].
Skip gram skip model [12] learns the vector representations which maximizes the probability of the word occurring in a context. A detailed explanation can be provided in Word2Vec Explained [7].



Challenges with Single Embedding

Interesting Observations:

- The above model would have been correct if each word had single meaning thus occurring in similar contexts. The polysemous words tend to be in different contexts which leads to having word representation which is average of word vectors for different contexts.
- The contexts corresponding to different senses of a word tend to remain in different clusters.
- The above model only considers local contexts. Many a times global contexts do provide significant observation, as shown in Huang et al. [9]
- Generally, the computed context vectors for different words can give an estimate of the sense of the word. This fact is exploited in all models aimed at finding multi word embeddings.

4 Multi Sense Vector Representations

To incorporate multi-sense vector we need to have sense vectors for each word. Slight modifications to the training function and cost mentioned in the previous section are needed to handle the Multiple Sense model.

4.1 Challenges with Multi-sense Word Vectors

- The number of sense for a word is unknown and dynamic. It's really difficult to estimate the number of senses which is the crucial part in obtaining multi-sense word vectors.
- To obtain context vectors from word vectors is again an approximation which might lead to error. Generally tf-idf weighing is done on a window having length of about 5 on each side.
- The context vectors and word vectors both are dependent on each other, since we need to know the corresponding sense of the context word before computing the context word.

4.2 Measuring Semantic Similarity

There can be many notions of similarity because of many senses of word. There are different variants of the cosine similarity in the multi-sense word vector models.
Let the words be w and w' and their contexts c and c' respectively. Let $P(w, c, k)$ denotes that word w takes k^th sense in the context c, and $d(v_s(w, i), v_s(w', j))$ is the single vector similarity measure. GLOBALSIM ignores the multiple senses and finds similarity between global vectors,

$$globalSim(w, w') = d(v_g(w), v_g(w')) \tag{1}$$

AVGSIM computes average similarity over all senses, thus it considers the existence of multiple senses. It's observed to be a good measure of word similarity. Notice that it doesn't consider contextual information.

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

AVGSIMC is the extension of AVGSIM which takes information of context into account. Instead of just taking average it weighs the similarity of words with the context by checking how well does the word fit with the context.

$$avgSimC(w, w') = \sum_{i=1}^{K} \sum_{j=1}^{K} P(w, c, i)P(w, c', j)d(v_s(w, i), v_s(w', j)) \tag{3}$$

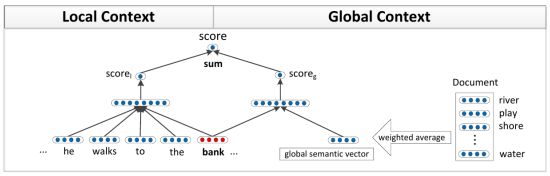
LOCALSIM selects the particular sense of the words using their contexts and computes similarity between them,

$$localSim(w, w') = d(v_s(w, k), v_s(w', k')) \tag{4}$$

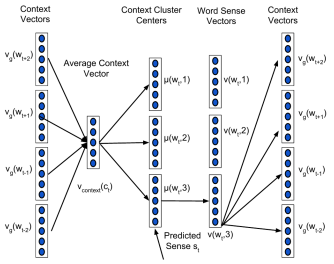
where k and k' are the predicted sense of the words respectively.

5 Previous Work

There has been a recent spur of interest in this field [5] [3] owing to its importance and fundamental nature. Work has been done showing that this problem can go hand in hand in with Word Sense Disambiguation [4]. We discuss the famous approaches to the problem. There has been relatively less work in the field considering the amount of attention given to Word Vectors.



(a) Usage Of Global Contexts [9]



(b) MSSG Model [14]

Figure 2: Previous Models Used

5.1 Parametric Sense Estimation

- **Using Local Contexts** : Risinger and Mooney [17] in 2010 presented the multi-prototype vector space model for lexical semantics with a single parameter which is number of clusters. They used clustering on local contexts to show that context vectors for different senses lie in different clusters. After clustering context vectors([17] uses feature vectors instead of context vectors) similarity between two word types are computed which is a function of cluster centroids, instead of centroid of all the occurrences.
- **Using Local Contexts with Global Contexts** : Huang et al[9] improved the model by incorporating global context with local context to improve the word representation to learn multiple dense, low-dimensional embeddings by using recursive neural networks.

5.2 Non Parametric Sense Estimation

Neelakanthan et al [14] have explored the non-parametric method for finding different number of senses for each word. Instead of learning context vectors as part of a pre-processing step and learning word vectors from it, [14] simultaneously learn word and context vectors because each of them were dependent on each other. The number of senses initially is unknown. Neelakanthan et al [14] followed an online approach for clustering during which they also estimated the number of clusters. This approach provided flexibility and different words could have different number of senses.

6 Approaches Used

The journey to find the best algorithm for the multi sense computation involved lots of detours and failures. We started off with online algorithms [2] and slowly moved on to other approaches.

6.1 Online Parametric Clustering

Our first attempt was based on online algorithms on clustering. We read about previous work done in online clustering [2]. There are many algorithms which give a strong theoretical bound on the "niceness" of the clusters. We formulate our algorithms on similar lines taking inspiration by Neel et al. [14] and other clustering algorithms.

The rough description of the algorithm used is as follows,

1. Each word $w \in W$ is associated with sense vectors, context clusters and a global vector $v_g(w)$
2. The number of senses for a word K is taken as a parameter, it is set to 3 in our experiments
3. Initially, the K sense vectors and context clusters are initialized randomly. The global vector is taken as an input or is computed using Word2Vec [1]
4. Given a word and its context, we compute a context vector. We then find the sense to which the word belongs to by computing the similarity between the context vector and each sense vector. We assign the most similar sense as the word's sense.

5. We also update the sense cluster/vector by adding the new word to it. The sense vector becomes the average of all the word vectors which belong to it.
6. Variations:
   A Iterate over the steps till convergence using some metric
   B Fixed number of iterations
   C Successive Approximations : At each iteration the context vector construction uses the sense (decided in the previous iteration). We hope that this helps in getting better context vectors.

We realized soon after that the results are similar to the standard K Means algorithm [20]. There are other improved online algorithms which give guarantees of "niceness".

6.2 Online Non Parametric Clustering

We have stressed on the importance of non parametric clustering in previous sections. The rough description of our algorithm which incorporates non parametric clusters is as follows,

1. Each word $w \in W$ is associated with sense vectors, context clusters and a global vector $v_g(w)$
2. The number of senses for a word is learned during training.
3. Initially, the K sense vectors and context clusters are initialized randomly. The global vector is taken as an input or is computed using Word2Vec [1]
4. A new context cluster and a sense vector are created online during training when the word is observed with a context where the similarity between the vector representation of the context with every existing cluster center of the word is less than λ, where λ is a hyperparameter of the model.
5. We have taken λ to be in the range $[-0.5, -0.4]$ for our experiments
6. The update operation is same as in the previous section
7. Variations: They are the same as in the case of parametric clustering.

Flaws And Challenges : We faced many difficulties in improving the results of the online algorithm. A majority of them were due to poor representations of contexts. We noticed that there was no inherent structure in context vectors of words, the corresponding TSNE plots [19] look extremely scattered. Better context vectors might be a possible solution.

Implementation Details:

1. Implementation Done in : C++, Python
2. Four Variants (Variants discussed earlier plus changes in context construction)

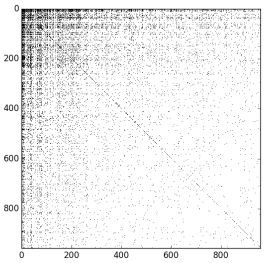
6.3 Word Word Co-occurrence Matrix

Moving on from online clustering methods, we also tried inferring information from the word word co occurrence matrix. The construction of the matrix is pretty intuitive. We consider a fixed size context window around every word and increment the corresponding entries in the matrix.

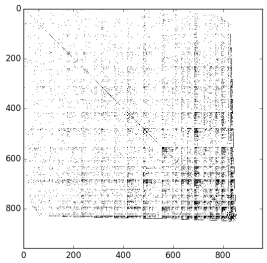
We use Reverse CutHill Mckee Ordering [11] to permute the words to get an idea of meaningful clusters. Unfortunately we weren't able to infer senses from it even though the visualization seemed helpful. We show the following results found with this approach,

Implementation Details:

1. Implementation Done in : Python
2. Scipy used for RCM ordering



(a) Word Word Co-occurrence Matrix



(b) Matrix after reordering using RCM [11]

6.4 Parameter Estimation

Disappointed from the failure in the word word co occurrence matrix we decided to focus on fundamental problems in our approach. Thus, we decided to focus on Non Parametric Models to target the problem.

An important constituent of the project is estimating the optimal number of senses. We tried to find ways in which this could be done. Along the way we encountered methods such as X-Means and Dirichlet Process Mixture Models [10]. There were many clustering algorithms which were able to handle uneven clusters which seems to model real life polysemy really well. Unfortunately, we decided to not use the clustering algorithms mentioned due to time constraints and also their time complexity.

We finally propose a method based on evaluation of cluster quality, which further depends on negative sampling. We discuss the method in more details in the following sections.

7 Final Approach

The main focus of our project changed during our journey through it. Slowly realizing the importance of this problem, we decided to focus on constructing an algorithm which acts similar to a black box and constructs a Multi Sense Vectors using original Single Word Vectors (possibly taken as input).

Hence, we make the computation of initial embeddings and recognition of multiple senses two independent tasks. Thus we simply feed in the embeddings and get the multi word prototypes.

Things We Know:

- Lots of work done for computation of better word representations
- Considerably less amount of work done in computation of multi word prototypes
- Non parametric computation almost non existent (We know of only one such paper)

. Which means:

- Creation of such a black box (which gives us the multiple senses) could easily improve the existing representations
- A 8-12% rise in spearman correlation for the SCWS task [9] has been seen Neel et al [14]

7.1 Methodology

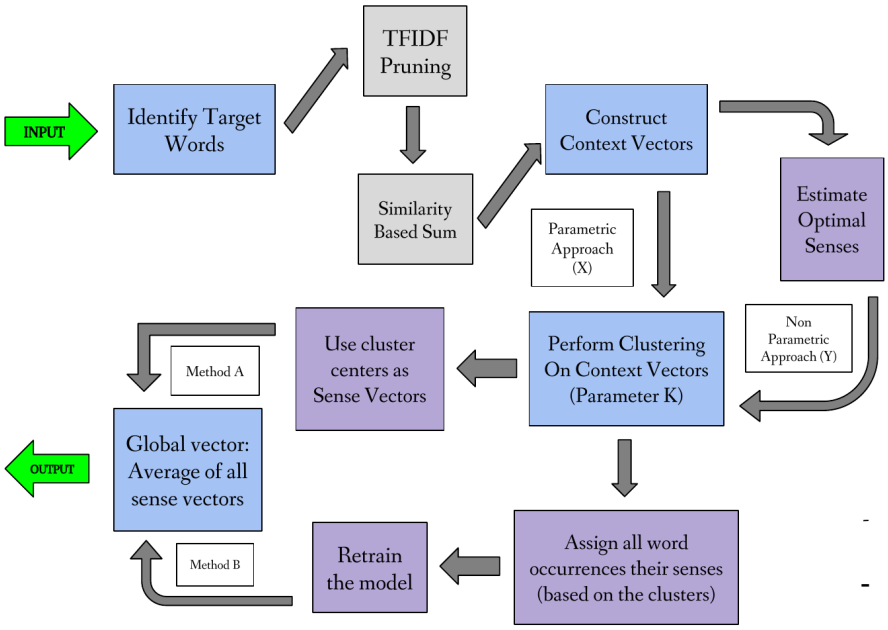
Following up on our black box agenda, we take single word representations. And use it as black box to create context vectors and then utilize it to construct sense vectors for each word. We give algorithms for both parametric and non parametric versions of the problem. The details are explained in further sections.

1. Single sense word embeddings as input OR Construction of word embeddings [3]
2. Identify top M words for which we compute multiple senses (Generally by frequency)
3. Construct context vectors:
    - TFIDF Based weighting: Intuition is to consider only influential words in the context
    - Similarity Based Weights: Pure heuristic; Consider words already similar to the curren t word; Induces bias
    - TFIDF Pruning: Completely remove "useless" words; Gets rid of noise in contexts
4. Estimate the optimal number of senses (clusters) OR Use the given parameter.
5. Perform clustering using the estimate.
6. Use cluster centres as sense vectors [1]

Further Improvements/Variants

- Approach A : 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]
- Approach B : Use the estimated sense vectors to assign senses to all occurrences of the words; Retrain using skip gram model; We get a different vector space compared to the original

The overall model can be seen in the following flow chart,



7.2 Parametric Estimation

The underlying assumption in the Parametric method is that the number of clusters i.e. number of senses for each word is a constant and is given by us. On average, number for senses for multi-sense word is 3 to 4. Consequently, it is the value we use for our experiments. Context vectors are clustered using K-means (where K is the parameter). We can proceed with approach A or approach B described above to obtain the Multi-Sense word vectors.

7.3 Estimation of Optimal Clusters

We have already stressed on the importance of non parametric clustering in previous sections. Many methods to incorporate varying senses have also been discussed earlier. In our project we provide a method based on the technique of Negative Sampling [7].

We define the probability of observing word c in context w_t as,

$$P(D = 1|v(w_t), v(c)) = \frac{1}{1 + e^{-v(w_t)^T v(c)}} \tag{5}$$

The probability of not observing word c in context w_t is given by

$$P(D = 0|v(w_t), v(c)) = 1 - P(D = 1|v(w_t), v(c)) \tag{6}$$

The modified objective function is given by

$$\begin{aligned} J(\theta) = & \sum_{(w_t, c_t) \in D^+} \sum_{c \in c_t} logP(D = 1|v(w_t), v(c)) \quad + \\ & \sum_{(w_t, c'_t) \in D^-} \sum_{c' \in c'_t} logP(D = 0|v(w_t), v(c)) \end{aligned} \tag{7}$$

Where c_t is the context vector and c'_t are randomly sampled noisy context vectors for w_t. D^+ consists of word-context pairs(positive sampling) and D^- consists of word-context pairs which do not occur together (Referred to as Negative Sampling)

The estimation roughly works in the following manner,

- Start with an initial guess on the number of senses; Initial value is one.
- Performing clustering based on the guessed value; Get the sense vectors for the computed clusters.
- Compute the cost function discussed earlier.
- Make a better guess based on the cost OR Accept the number of senses; Current implementation makes another guess until the improvement in the cost is not substantial

7.4 Non Parametric Estimation

We know that the assumption made by Parametric models are grossly incorrect for real life data. Thus, Non Parametric Model has been our main focus in the project. We aim at building a model to handle variable number of clusters for reasons discussed earlier.

Estimation for the optimal number of clusters is performed for every word using the approach discussed in the previous section. The process flow can be seen in the Flow chart shown earlier.

8 Datasets

- Stanford's Contextual Word Similarities (SCWS) [9]: Consists of a pair of words, their respective contexts, the 10 individual human ratings, as well as their averages
- Training Corpus [18]: Same as the one used in Huang [9] which is april2010 snapshot of the wikiepedia corpus(Shaoul and Westbury,2010). It contains approximately 2 million articles and 990 million tokens.

9 Results

Deciding whether the computed Multi Sense Vectors is an extremely difficult task and thus we can not claim anything just based on comparisons on given data sets. We provide results and comparisons for some common tasks which may help us decide the correctness of our embeddings.

9.1 Nearest Neighbours

Nearest Neighbors are great way to compare word embeddings since we can get an idea of the relationships between words in the computed Vector Space. We give the results based on Y B algorithm computed on the training data described above.

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

Table 1: Nearest Neighbors

We see that our model correctly identifies different senses and can also incorporate variable number of senses for different words.

Model	Nearest Neighbors (For Word : Plant)
Skip Gram	plants, flowering, weed, fungus, biomass
MSSG [NEEL]	plants, tubers, soil, seed, biomass, refinery, reactor, coal-fired, factory, smelter, asteraceae, fabaceae, arecaceae, lamiaceae, ericaceae
NP MSSG [NEEL]	plants, seeds, pollen, fungal, fungus, factory, manufacturing, refinery, bottling, steel, fabaceae, legume, asteraceae, apiaceae, flowering, power
Huang Et Al [2]	insect, capable, food, solanaceous, subsurface, robust, belong, pitcher, comprises, eagles, food, animal, catching, catch, ecology, fly, seafood, equipment, oil, dairy, manufacturer, facility, expansion, corporation, camp

Table 2: Nearest Neighbors For Other Models

We see that the discovered senses in both our models are more semantically coherent than the other models We our also able to learn reasonable number of senses. We suspect this to be due to formation of extremely fine senses in the other models.

9.2 Evaluation on SCWS Task

We use Stanford's Contextual Word Similarities (SCWS) as the base for comparing different models. The main reason being the presence of word contexts. It consists of a pair of words,

their respective contexts, the 10 individual human ratings, as well as their averages.

We compare all of our models with other models approaches on different notions of similarity [6]. The scores are obtained by computing Spearman correlation between our predictions and the human ratings (multiplied by 100).

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

* Unable To Recreate Score

Table 3: Results On SCWS Task

Both our parametric and non-parametric models are comparable to the state of the art models. We are able to outperform all present models in LocSim measure. The best results according to each metric are in bold face.

9.3 Execution Time

Training time is also an important feature in deciding the effectiveness of a given model. We describe our results table 4.

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

Table 4: Training Time

It takes reasonable time to produce the results. Our models are much faster than previous models and lose only to Neel et al's [14] model. Also since the code is written in python, it's expected to

have 3-4 times faster runtime in C++.

9.4 Discussions

It is interesting to see that our model can outperform state of the art algorithms in the LocSim measure. A possible implication is that our model actually learns the optimal number of senses for a word; while other models "may" be creating fine clusters (senses) leading to poor performance with LocSim measure. Experiments to check the hypothesis that our model actually disambiguates to the exact sense would indeed be desirable. It would be interesting to compare the formed senses with a present sense bank (Possibly WordNet [13]).

Since we are able to compare to the other models using other similarity measures, our model can serve as an efficient pre processing step in many tasks i.e. use it to get Multi Sense Embeddings, further helping in getting better results.

10 Applications in Other Course Projects

The fundamental nature of the task makes it capable of improving results in many tasks. We mention some of the course projects which we feel can get better results using Multi Sense Word Embeddings.

- Character Word Embeddings for NLP tasks in Indian Languages : The parent paper of the project contains non parametric embeddings i.e. multiprototype character embeddings; its formulation is extremely similar to our own work.
- Diachronic Word Sense Change Identification : The project aims at finding the change in the usage of different words over time. Results might be worsened due to presence of multiple senses and thus identifying different senses from the get go can be helpful in improving the final analysis.

11 Future Work

- Since clustering is an important constituent of our proposed algorithms, we should aim at improving it. The possible methods to improve quality of clustering:
  - A Improved Non Parametric Clustering: Employ a clustering model with infinite capacity, e.g. the Dirichlet Process Mixture Model [10]. Models such as X-Means [15] can also model variable number of clusters. These allow more polysemous words to adopt more representations. Tackles the issue of varying word senses.
  - B Improved Quality Of Clusters: Many clustering algorithms try to compute clusters of even size. Which is an incorrect assumption to handle real life polysemy, since many senses of words are used different number of times.
- Using Multi Lingual Corpora to help improve Sense computation [8]. There has been a course project related to this work.
- Cluster similarity metrics: Other similarity metrics over mixture models [6], 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.

12 Bibliography

References

[1] Word2vec. `https://code.google.com/p/word2vec/`.

[2] Wesam Barbakh and Colin Fyfe. Online clustering algorithms. *International Journal of Neural Systems*, 18(03):185–194, 2008.

[3] Tao Chen, Ruifeng Xu, Yulan He, Xuan Wang, et al. A gloss composition and context clustering based distributed word sense representation model. *Entropy*, 17(9):6007–6024, 2015.

[4] Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035, 2014.

[5] Jianpeng Cheng and Dimitri Kartsaklis. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. *arXiv preprint arXiv:1508.02354*, 2015.

[6] James Richard Curran. From distributional to semantic similarity. 2004.

[7] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[8] Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. Learning sense-specific word embeddings by exploiting bilingual resources. In *Proceedings of COLING*, pages 497–507, 2014.

[9] 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.

[10] Sinae Kim, Mahlet G Tadesse, and Marina Vannucci. Variable selection in clustering via dirichlet process mixture models. *Biometrika*, 93(4):877–893, 2006.

[11] Wai-Hung Liu and Andrew H Sherman. Comparative analysis of the cuthill-mckee and the reverse cuthill-mckee ordering algorithms for sparse matrices. *SIAM Journal on Numerical Analysis*, 13(2):198–213, 1976.

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[13] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[14] 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.

[15] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734, 2000.

[16] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543, 2014.

[17] Joseph Reisinger and Raymond J Mooney. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics, 2010.

[18] Cyrus Shaoul. The westbury lab wikipedia corpus. *Edmonton, AB: University of Alberta*, 2010.

[19] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[20] Shi Zhong. Efficient online spherical k-means clustering. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 5, pages 3180–3185. IEEE, 2005.