CREATING SENSE VECTORS WITH CROSS-LINGUAL DATA

Deepanshu Gupta, Rachita Chhaparia

Guide: Dr. Amitabh Mukerjee

CS671A: Introduction to Natural Language Processing Indian Institute of Technology, Kanpur

Introduction

Today, most word representation models assume a single representation of all words which is problematic owing to the existence of polysemous words[5]. Therefore, the task is to create sense vectors for polysemous words i.e. a set of distinct vectors to represent each sense.
In our project, we create and use sense vectors to approach the problem of Word Sense Disambiguation.

Our Approach

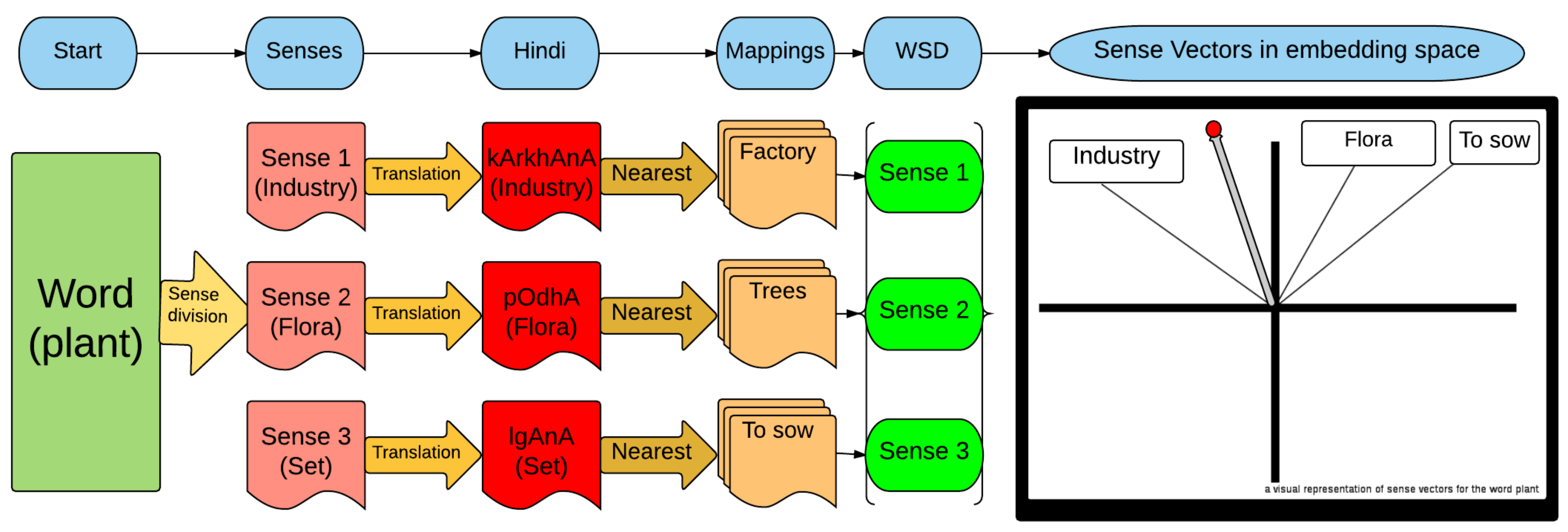
- Distributed representations of similar words in two languages are related by a linear transformation as shown in [4]. For our problem, we learn the linear mapping from the Hindi to the English word vector space.
- For a polysemous word in English (Hindi), we translate its synset in Hindi (English). We map the vectors of each translated word in the synset back to the English (Hindi) vector space using the matrix learnt above which are the corresponding sense vectors.
- For WSD, the sense whose vector has the maximum cosine similarity with the context vector is selected.

Methodology

1. **Word Vectorisation**: English: Pre-trained vectors (about 1B words) using Mikolov's Word2Vec with continuous bag of words architecture. The model contains 300-dimensional vectors for 0.2 million words.
   Hindi: Vectors trained on HindMonoCorp0.5 using Mikolov's Word2Vec with skip-gram architecture. The model contains 300-dimensional vectors for 0.07 million words.
2. **Learning the Translation Matrix**: A linear mapping i.e. a translation matrix $W_{h \to e}$ from the Hindi word-vector space to the English word-vector space was learned using stochastic gradient descent using a bilingual dictionary consisting of most frequent 5000 words in the Hindi corpus for training (obtained using online Google Translate).
3. **Sense Translation**: The senses of 25 polysemous words in English were translated into Hindi using online Google Translate.
4. **Context Vectorization**: A context window of size, say 'k' (a parameter) was taken around the target word and their vectors were averaged to form the context vector, We also, down-weighted the vectors based on their frequency in the corpus and their distance from the target word in the context.
5. **Sense Vectorization**: For every English polysemous word, the vectors (x) of its translated senses are mapped back to English ($W_{hi \to en} x$) which are the corresponding sense vectors for the word.
6. **Sense Disambiguation**: For every target word, we calculate the cosine similarities between the context vector and each of its sense vectors created above. The sense whose vector has the maximum cosine similarity is chosen as the correct sense.

Implementation Structure

Consider the disambiguation of the polysemous word **plant**:



Experiments

The test set was taken from Semeval-I, consisting of approximately 200 instances each of 30 polysemous words. The synsets were made coarse-grained manually. The average accuracy across all instances was **42%**. Some examples:

Phrase	Correct	Prediction
unable to work because of unemployment, illness or an **accident**.	crashmod	crashmod
they have the **sanction** of the New Zealand Rugby Union for a simple stratagem	econ- action	econ- action
There was silence then while we extricated the **scraps** of crisp batter from the folds of the paper.	morsel	waste
Hitting an industry when it is nearly on its **knees** is timely research.	bow	patella

Results

Results for some words:

Words	Accuracy (simple vector average)	Accuracy (similarity weighted average)
giant	90%	91%
amaze	88%	95%
promise	72%	73%
derive	55%	56%
sanction	55%	55%)
modest	36%	37%

Insight

- The accuracy for a word depends on the sense translations made. The more distinct they are, the better is the accuracy. For example, for the word 'knee', consider two senses 'bow' and 'patella'. They are used in almost the same context in both languages and hence, cannot be disambiguated accurately (around 50%)
- Changing the way we calculate the context vectors has a positive impact on the prediction task. We can also explore other context measure to improve the WSD predictions.

Conclusion

Creating sense vectors with cross-lingual information and using it for the problem of word sense disambiguation gives very accurate results when the translated senses are very distinct. Also, similarity measured with the context vector formed after weighting down words depending on their frequency and distance from the target word gives more accurate results.

Future possibilities

- Extension to phrasal embeddings
- Using non-linear or neural language models to generate translation function
- Reducing the hubness property

Toolbox and Resources

We used:
1. Google news vectors, Hindi Corpus[1]
2. Word2Vec[3]
3. Learning Translation Matrix [3]
4. Senseval-1 Test Data [2]

The source code and data sets are available at:
home.iitk.ac.in/~dpanshu/cs671/project/code.zip

References

[1] Ondrej Bojar, Vojtěch Diatka, Pavel Rychlỳ, Pavel Straňák, Vít Suchomel, Aleš Tamchyna, Daniel Zeman, et al. Hindencorp–hindi-english and hindi-only corpus for machine translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, 2014.

[2] Adam Kilgarri. Senseval: An exercise in evaluating word sense disambiguation programs. In *Proc. of the first international conference on language resources and evaluation*, pages 581–588. Citeseer, 1998.

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

[4] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[5] 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.