CIS 520 Spring 2018, Project Proposal

Word Sense Disambiguation by Supervised Learning

Team Members:

Haoxian Chen (PennKey: `hxchen`; Email: `hxchen@seas.upenn.edu`)
Leshang Chen (PennKey: `leshangc`; Email: `leshangc@seas.upenn.edu`)
Hui Lyu (PennKey: `huilyu`; Email: `huilyu@seas.upenn.edu`)
Yanci Zhang (PennKey: `yanci`; Email: `yanci@seas.upenn.edu`)

Assigned Project Mentor:

Anant Maheshwari

Team Member Contributions:

Team Member	Contributions
Haoxian Chen	Preprocess Data Experiment Naive Bayes classifier
Leshang Chen	Experiment logistic regression Plot results
Hui Lyu	Experiment SVM Preprocess Data
Yanci Zhang	Experiment Neural Network

Code Submission: GitHub: https://github.com/HaoxianChen/520-project

Abstract

1 Introduction

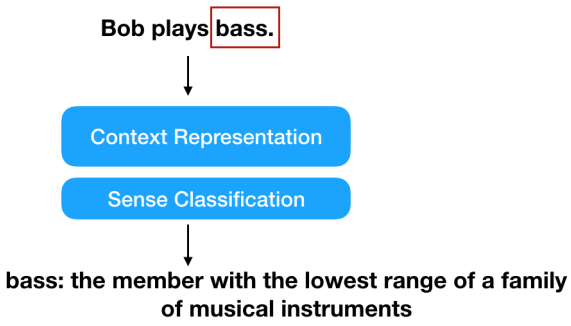


Figure 1: system overview

Words in natural language could have multiple meanings. Though humans are good at inducing the actual meaning of words given a certain context, such multi-meaning words are often seem ambiguous to machines. Such ambiguiaty in natural languages have been the major challenges for machine to understand user intention, e.g. google search key words.

To solve this problem, we propose to build a word sense disambiguate system that takes an English word and a sentence where it appears as input, and predicts the sense of such word within the given context. The architecture of our system is shown in figure 1.

2 Related Work

Major approaches to the word sense disambiguation problem can be briefly categorized into three types: Knowledge-based, supervised learning, and unsupervised learning.

Knowledge based appraoch. Lesk's algorithm [1] is a dictionary-based approach. It assumes that words appear within the same context have related semantics. The alogrithm disambiguate word sense by finding the dictionary definition that shares the most common words with the given context sentence.

Supervised learning approaches. These approaches typically use a hand-annotated Corpus as the training data set. The classifier is concerned on each individual word, and the output is one of the definition of the ambiguous word in the dictionary. Over the decades, people have proposed different algorithms to solve this classification problem: decision-tree [2], Naive Bayers classifier [3], Neural Network [4], and SVM [5], etc.

Unsupervised approach. Such approaches [6] assumes that similar senses appear in similar contexts. By clustering occurrance of words into clusters using some similarity measures, the sense of a new occurring word can be induced by assigning it to the closest cluster.

3 Dataset

We use Semcor [7] to generate our dataset. Semcor is a Corpus that consist of a set of tagged sentences. The tags provided with each word. The tag information we used in this work includes:
(1) Word sense ID in WordNet [8], which is used as prediction label.
(2) Part-of-Speech tag. This is used to encode context.
(3) Lemma. This is used to remove the inflections of word form variations in different contexts.

Specifically, in our work, we use lemma, instead of the word itself, as the identifier for word occurrence counting. However, in context representation, we use the original word.

Class imbalance is prevalent in ambiguous words. Most words have some rare meanings with very few occurrences in the corpus. This ends up minority class in the dataset: classes that only appears in a few times (typicall less than ten times).

We discuss how we account for the class imbalance issues in our evaluation (section 6.3).

4 Problem Formulation

We formualte the word sense disambiguation task into a supervised multi-class classification problem. This problem can be represented as tuple (X, Y, h), where X is the instance space, that contains the description of the object on which we are to predict, and Y is the label space, from which each object draws a label, and h is the classification function $X \to Y$, that maps the descriptoin of an object in instance space X to label space Y.

Each targetted ambiguous word w has it's own label space Y_w, and a shared instance space X with other ambiguous words. Therefore, the solution to a word sense disambiguation is specific to each individual ambiguous word. For each ambigous word we train a specific classifier $h_w : X \to Y_w$, that maps instance from X to it's own label space Y_w.

4.1 Data Preprocessing

We use SemCor Corpus [7] to provide the raw data for human languages. SemCor Corpus is a corpus where each word is tagged with a sense provided by WordNet [8]. We then generate the dataset for this problem, in the following steps:

1. Tokenization: transform the sentence strings into list of tokens (usually words).
2. Part-of-Speech tagging: e.g. "the/DT bar/NN was/VBD crowded/JJ", where DT stands for Determiners, NN stands for Noun, etc.
3. Lemmatization: e.g. $plays \to play$, $was \to be$.
4. Feature extraction: select some features to represent the context.

Ambiguous word. Given an English word w, and a tagged corpus, we say the w is ambiguous, if it is tagged with at least two different WordNet Sense IDs in the corpus.

Context Representation We use Colocation [9] method to represent the context of a word. For a given ambiguous word, we looks at the preceding and succeeding words, and their Part-of-Speech tags. The size of the moving window can vary, we use two in this project, as suggested by previous studies (CITE).

$$[w_{i-2}, POS_{i-2}, w_{i-1}, POS_{i-1}, w_{i+1}, POS_{i+1}, w_{i+1}, POS_{i+1}] \tag{1}$$

We use Word2Vec [10] to transform each word (originally presented as string) into float-point vectors. Word2Vec [10] is a word embedding models that generates vector representations for each word in a given Corpus. The output vectors encodes the similarity among words in the context

of the training Corpus. We set the vector dimension for each word at 100, as suggested by the documentation of Word2Vec.

To generate vector representation for POS tags, we use Word2Vec embedding model again, but substitute the words with its POS tag when training the model.

Construct training set for each target word. Let W be the set of English words that we want to do sense disambiguation. For each $w \in W$, we need to extract a training set X_w from the SemCor Corpus as follow:
(1) Draw a set of sentences S_w from SemCor Corpus, where each $s \in S_w$ contains w.
(2) Convert each sentence $s \in S_w$ to a feature vector x, and the sense label y. The set of all (x,y) pairs is our training set X_w.

5 Model and Algorithms

For this multi-class classification problem, we consider the following models, using one-vs-all method to extend them to solve multi-class problem:

Naive Bayes

Logistic Regression We use the Logistic Regression package provided in scikit-learn library and runs the experiment in python.

Support Vector Machine
We have built three types of SVM models for each target word: Linear SVM, RBF kernel SVM and Polynomial kernel SVM. For the purpose of multi-class classification task, we use the `svm.SVC()` method in `sklearn` module in Python. Specifically, we set the `kernel` attribute in `svm.SVC()` to be "linear", "rbf" and "poly" respectively for building the three types of SVM models.

Neural Network Since the feature contains the sentence structure, the model is designed to process having capability of recognize which word come first and later instead up allowing model randomly search the best combination. Causal convolution layer [11], a type of convolution, has been used in this problem to maintain this time series like structure. In order to having a larger size of reception field, dilation blocks are used, in which each layer extract certain amount of feature with sequence information [12], so that it is quite possible to process a much longer sentence without changing the structure with additional information. Residual network structure [13] has been used on dilation block to prevent gradient vanishing in deep network.
The overall model structure is a forward causal layer to process sentence in a forward sequence, followed by two dilation residual blocks. There is a parallel structure to process the sentence backward. Two dilation blocks skip connections on both sequence are concatenated, then into a fully connected layer and enters a Softmax layer to make classification prediction.

Desired properties. Since in our system design, we need to train a classifier for each target word, the ideal method should have short training time, and compact model representation.

6 Experimental design and results

6.1 Dataset Selection

There are more than thousands of ambugious words in the SemCor corpuse, by our definition in secton 4.1, which will translate to more than thousands of datasets and the corresponding machine learning tasks. However, not all dataset is qualified for a tractable machine learning problem. The problem is that some words have senses that are rarely appeared in the sencor corpus, which is typically 1 or 2 appearances. The lack of data makes the word sense unlearnable. Therefore, we use two criteria to select the set of training set we use in our experiment: (1) Overall instances greate than 200, and (2) Each sense ids (labels / classes) appear at least 20 times.

6.2 Parameters Tuning

Logistic Regression We first perform a parameter search over 5-fold cross-validation over the entire training set for different words. We tested different regularization values, solvers, multiclass metrics. Finally we found that in most cases by setting C=1, solver="newtown_cg", max_iter = 500, and metric="multinomial" we can get the best accuracy and f1 scores. We'll set them to be the global parameters over logistic regression classifier for each word.

SVM. For each type of SVM models, we execute a 3-fold cross-validation on the training set for choosing the best parameters. The `grid_search.GridSearchCV()` method in `sklearn` module in Python is used for this. We maintain the default stratified 3-fold splitting approach during cross-validation. The parameter to be tuned for Linear SVM is C, for RBF SVM is {C, gamma}, and for Polynomial SVM is {C, degree}. The range of values for each parameter is set as follows.

C = [0.01, 0.1, 1, 10, 100]
gamma = [0.0001, 0.001, 0.01, 0.1, 1, 10]
degree = [2, 3, 4, 5]

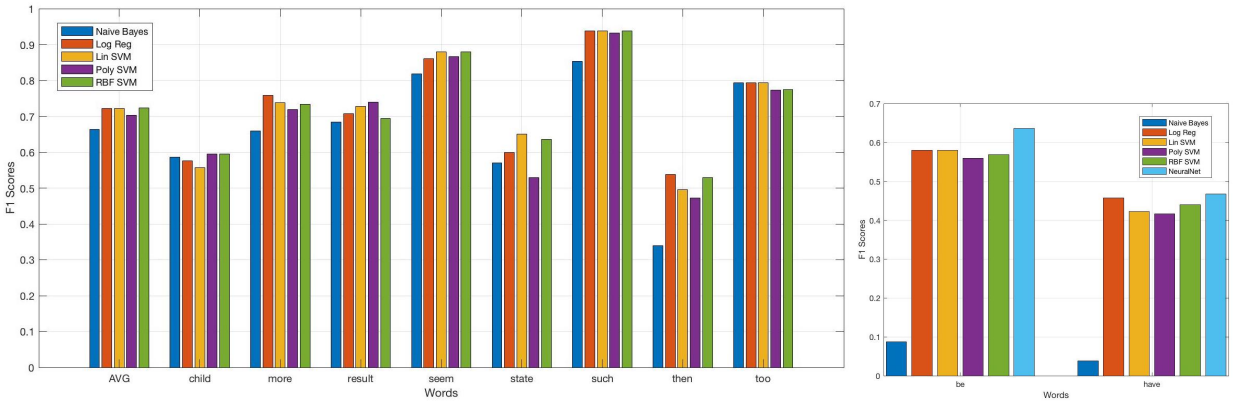
Eventually, for each target word, each type of SVM models has its own tuned best parameters.

Neural Network. Because the limitation of dataset and purpose of in comparison with other modeling method, the model parameters used in the network for comparison are two dilation blocks with each only two layers to process four words. Only forward processing blocks exist, since the dataset is too small, which has been under-fitting. Number of filters of both causal convolution and dilation layers is 8. Number of filters of dense layer before the softmax layer is 16. Adam was used as the optimizer. The data is trained the whole dataset of a word per batch, and repeated five times for the same dataset.

6.3 Results: Weighted F1 Scores

We use **weighted** F1 scores to account for the class imbalance effect in the dataset. That is, we first calculate F1 scores for each class, and then average them, weighted by the number of each class in the test set.

Using the criteria in section 6.1 we filter out the eight words in evaluation result in figure 2a. But these dataset consist too few training instances for our neural network algorithm to converge. To evalute the neural network's performance, we loose the second criterion (fewer instance on some



(a) Weighted F1 scores on small-scale dataset

(b) on larger dataset

minor classes) and include two larger dataset, and results in figure 2b. We handle the minor class issues (classes with only one or two labels), by manually upsampling the instances of such classes to 20.

Logistic Regression and SVM works consistently well Comparing the bars for Logistic Regression and SVMs, in figure 2a and figure 2b, we notice they yields close F1 socres on both small and large dataset.

Kernel method doesn't give significant improvement to SVM. Neither RBF Kernel SVM and Polynomial Kernel SVM shows better performance over linear SVM.

Neural Network dominates large dataset. Even though the part of the dataset is synthesized, (upsampling to handle class imbalance), Neural Network outperforms the rest of the classifiers.

Naive Bayes only works at small-scale dataset. A significant performance downgrade shown on large dataset (figure 2b). However, on small dataset, Naive Bayers is able to catch up with the other discriminative models.

7 Conclusion

We solve the word sense disambiguation problem using supervised learning methods, We gather training data by generate feature instances and labels from hand-tagged Corpus with word meaning defined by WordNet. We evaluated multiple multi-class classification medels and algorithms on this dataset, the result shows that Logistic Regression is the best all-around algorithm to solve this word-sense classification problem, considering training time, model complexity, and the overall performance based on F1-score.

Neural network underfit issue. Because the dataset is very small, the simplified neural network model remains severely underfit. More data is required to train the model in order to improve the performance instead of adding more features into the dataset. The network was originally designed for the whole sentence processing, so as long as given a sufficient dataset, there is a huge potential to give a better result. For future work, each dilation block could be replaced by LSTM, which is also proven strong at time series type data processing.

References

[1] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM, 1986.

[2] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[3] Hwee Tou Ng. Getting serious about word sense disambiguation. 1997.

[4] Garrison Weeks Cottrell. A connectionist approach to word sense disambiguation. 1985.

[5] Yoong Keok Lee and Hwee Tou Ng. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 41–48. Association for Computational Linguistics, 2002.

[6] Hinrich Schütze. Automatic word sense discrimination. *Computational linguistics*, 24(1):97–123, 1998.

[7] Semcor corpus. `https://www.gabormelli.com/RKB/SemCor_Corpus`.

[8] About wordnet. `https://wordnet.princeton.edu/citing-wordnet`.

[9] Colocation. `https://en.wikipedia.org/wiki/Collocation`.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[11] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[12] Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.