COMP 550 Assignment 3

Yves Blain-Montesano 260745418

November 12, 2018

Question 1

a) 1. $(\lambda x.xx)(\lambda y.yx)z \equiv (\lambda y.yx)(\lambda y.yx)z \equiv (\lambda y.yx)xz \equiv xxz$
   2. Answer: a
   3. Answer: $(\lambda v.w)$

b) Det → a { $\lambda P.\lambda Q.\forall x P(x) \implies \neg Q(x)\}$
   Answer: $\forall x Student(x) \implies \neg\exists e Hates(e) \wedge Hater(e,x) \wedge Hatee(e, COMP550)$

c) If 1 > 2:
   2. $\exists y Exam(y) \wedge \exists e Want(e) \wedge Wanter(e, s_1) \wedge Wantee(e,y)$
   1. $\forall x Student(x) \implies \neg(\exists y Exam(y) \wedge \exists e Want(e) \wedge Wanter(e,x) \wedge Wantee(e,y))$
   If 2 > 1:
   1. $\forall x Student(x) \implies \neg(\exists e Want(e) \wedge Wanter(e,x) \wedge Wantee(e,s_2))$
   2. $\exists y Exam(y) \wedge (\forall x Student(x) \implies \neg\exists e Want(e) \wedge Wanter(e,x) \wedge Wantee(e,y))$

Question 2

Setup

Various methods are compared for word sense disambiguation on a portion of the SemEval 2013 Shared Task #12. First, a baseline is used which disambiguates by assigning the sense label of the top-ranked synset taken from NLTK's WordNet interface. Next, the NLTK implementation of Lesk's original algorithm is used. Two unique methods are then used. The first incorporates a trained classifier into Lesk's algorithm. The second is an extension of Lesk's algorithm which compares an extended set of tokens. All (except the baseline) are compared with and without the exclusion of synsets with the wrong part of speech from the potential choice of sense. English stop words and any words under 2 characters of length were removed. The `@` character was removed due to its appearance in the word `@@card@@`. There were seemingly no other such symbols which would obscure a token from its true type. It is important to note that all POS tags in the set of instances given were nouns.

Methods

Extended Lesk's Algorithm

The original Lesk's algorithm compares the size of the intersection of the context tokens with each of the target word's potential sense definitions' tokens (here taken from NLTK's `Synset` class' `definition()` method). Here, the size of the intersection is that of the context and the synset's definition augmented by the intersection of the context and each of the synset's definition words' own synset's definition (baseline used). Then, the intersection of the target synset and each of the context's words' own synset's definition (i.e. the same as above, but "flipped") is added, as well as the intersection of each context word's definition with each of the target synset's definition's definition. Simply put, it is similar to Lesk's algorithm, though applied to a lower depth level and between depth levels.

Semi-supervised Lesk's Algorithm

This modified extends the above method. It weights each intersection size by the inverse of the complement of a classifier's prediction probability for that sense. The intuition is that although the raw numbers are useful in disambiguating, some words may have more weight for one sense than another. For simplification, the probability weights the whole intersection count rather than each word based on its importance for a sense.

To achieve this, a Gaussian Naive Bayes classifier is trained for each lemma in the development set where the input is the TF-IDF-vectorized matrix of the contexts of all instances of a given lemma and the class label is any of the `name()` strings of the `Synset` returned by `nltk.corpus.wordnet.synset_from_sense_key(str sense_key)` for each occurence of the lemma in the data set (including multiple senses). This label is automatically encoded when fitting the `GaussianNB` classifier provided by the scikit-learn 0.20.0 library.

Results

Below is an accuracy table for each method. Only the semi-supervised Lesk's Algorithm need be compared between the development and test set, though all scores are listed where applicable. Below is an accuracy table for each method. Only the semi-supervised Lesk's Algorithm need be compared between the development and test set, though all scores are listed where applicable.

	Dev	Test
Baseline	0.505	0.494
Lesk	0.222	0.214
Lesk w/POS	0.258	0.268
Ext. Lesk	0.268	0.295
Ext. Lesk w/POS	0.309	0.324
Semi-sup. Lesk	0.288	0.295
Semi-sup. Lesk w/POS	0.330	0.324

The baseline performs surprisingly well, although the rough nature of the methods implemented for this report makes this unsurprising. Nevertheless, they are a clear improvement over the original Lesk's algorithm. The heuristic methods used obviously could not benefit from the data provided, though the simple filtering of potential senses by part of speech is relatively effective. The extended method performs marginally better and is also improved by POS. It is possible that this is simply a case of manually overfitting without the use of a validation set.

As for the frequency-distributional property of the senses, too little of it seems to have been captured by the semi-supervised method to affect accuracy meaningfully. The reason for this is the use solely of the data provided. The use of a large corpus of training data would allow to classify more effectively between senses given a lemma. As it is, the models trained are restricted only to the classes encountered in training data, severely limiting their capacity to weight candidate senses' context intersection sizes. Moreover, a model could be considered that would more faithfully represent the classification problem at hand than one with strong independence assumptions like Naive Bayes models.

Perhaps this problem could be considered downstream from some information extraction task, such as named entity recognition, term extraction, or some semantic analysis, which all seem reasonably motivated by theoretical linguistics. It would likewise be interesting to explore mapping to senses as a sequence-to-sequence problem for mapping context to a sense key, given enough data.