NLP HW2 report

Jary Pomponi

I have choose a knowledge based approach based on graph with a novelty to accomplish this homework. The graph is as the ones used in literature: $G = (V, E)$ is an undirected graph where V are synsets and E are edges between two vertexes if exists a semantic connection between them; i have used NetworkX to build and manage the graph. Usually in literature the graph is build from a train set in unsupervised way: for each ambiguous lemma all of the possible synsets are added to the graph. My approach is a supervised one: for each ambiguous lemma i have added only the correct synset and all the ones that are connected to that one. I will call that $Gsup$. To do that i have used BabelNet to retrieve all the associated synsets and semantic connections between them.

I have done experiments with two different kind of $Gsup$. In the fist set of experiments all the edges had same weight, $Gsup_{nw}$, and in the second one i have used a co occurrence matrix to assigning edges' weight in order to encode strongly the connection between synsets in the train set, $Gsup_w$. To build the matrix i have used a context window on a documents fashion. We will see that both of approaches overcome baseline.

For each graph i have used different prediction approach, all based on pagerank[1] Before use any technique a subset of lemmas from dev set should be added to the graph. To do that we retrieve from BabelNet all the synsets associated to each lemma, but only the WordNet subset to avoid noisy data, with the help of pos tag, in the subset. Then all possible synsets for each lemma are added but edges are added only if the target node already belong to the graph. Selection of subset is the main difference between the prediction approaches. Approach to choose the most probably synsets are the same for all the approaches: pagerank vector are computed and the for each possible synsets of a given lemma the most probable one are picked. Usually the initial probability are placed on the synset generated from a subset of lemmas; the only exception are the static algorithm where all the nodes has same initial probability.

in the first one, static prediction S_p, add all the eval set in the graph and then compute pagerank vector used to disambiguate lemmas. In the mass version probability are placed on synsets appearing in eval set while the static approach give the same probability to each node in the graph. Second approach, document prediction D_p, are equals to the first one with mass probability, but instead of adding all the dev set to the graph we add only current document. In the third one, bfs prediction $SubBfs_P$, we add all the document as in the second one but then, for each synset associated to a lemma, we get all the closest synsets in a path that do not exceed a limit called cut using Dijkstra algorithm. The we extract a subgraph formed by nodes found with Dijkstra algorithm and all edges between them. Then pagerank vector is computed on that subgraph followed by the prediction for current lemma. This approach is slow and i have implemented a cache in order to speed up computation and avoid calculate the neighbor of a synset more times.

[1] All of them can be found in graph.py file. Some not working or not good unused approach, not documented, can be found in not_used.py

1 Figures and Tables

Table 1: Results on eval dataset

Graph used	Prediction algorithm	senseval2	senseval3	semeval2007	semeval2013	semeval2015
$Gsup_{nw}$	Static	52.89+0.89	46.70−1	36.92 +1.92	51.03+	51.56
	Static Mass	52.45+	46.75+	37.58+	51.27+	51.17+
	Document	53.59+	47.62+	39.12+	54.44+	53.71+
	Subgraph (cut=2)	53.85+	47.24+	38.90+	50.72+	51.76+
$Gsup_{w}$	Static	58.98+	56.32+	45.05+	59.24+	58.31+
	Static Mass	58.72+	56.05+	43.29+	59.30+	58.41+
	Document	59.81+	56.75+	45.71+	60.40+	62.42+
	Subgraph (cut=2)	**60.21**+	**57.08**+	**46.37**+	**61.55**+	**62.42**+
Baseline		52	47.7	35	49.	51.2