WORD SENSE ANALYSIS

Jeffrey How - [redacted]

COMP6781

Table of Contents

Experiment Background

The purpose of this document is to provide the reader with a description of the attached word sense disambiguation program. Additionally, an analysis of its experimentation results will be provided.

Basic Setup

- **Data Set**: Senseval-3 English Lexical Sample Task
- **Words**: add & bank: Chosen due to higher sample size. expect: Chosen for granularity.

Lexical Unit	Senses	Granularity[1]	Training Size	Test Size
add.v	6	Fine	263	132
bank.n	10	Fine	262	132
expect.v	3	Coarse	156	78

- **Classification Algorithm**: Naïve Bayes (self-implemented)
- **Default Parameters**:
    - Context Window Size: +/- 4
    - Stop-Word Filtering: see *Input Files & Tools* below
    - Smoothing: +1

Input Files & Tools

- *Senseval-3 English Lexical Sample*[2]
    - inputFiles\\EnglishLS.dictionary.xml
    - inputFiles\\EnglishLS.dictionary2.xml
    - inputFiles\\EnglishLS.train
    - inputFiles\\EnglishLS.test

[1] Table 1 from http://acl.ldc.upenn.edu/W/W04/W04-0807.pdf
[2] http://www.senseval.org/senseval3

    - inputFiles\\EnglishLS.test. Key
- Stop words: inputFiles\\common-english-words.txt[3]

Output

- outputFiles\\output_[***word***].txt: program's classifications of test set.
- outputFiles\\analysis_[***word***].txt: program's analysis of classifications compared to key.

Programming

The Java program provided consists of the following three classes:

Sense Statistic

A SenseStatistic is an object that encapsulates the data for a particular word sense. If a single word has *x* senses, it will have *x* SenseStatistics. A SenseStatistic's attributes (which are mainly populated during training) include the following:

- A tree **map** of all **context words** (words within the context window) from all training examples and their respective frequencies
- Total **number** of **occurrences** of sense
- Total **number** of the sense's **context words** in its training examples

Classifier

The Classifier class has contains three steps:

1. **Train**: Reads the Senseval training file and populates a hash-table of SenseStatistics with context word and frequency data.

[3] http://www.textfixer.com/resources/common-english-words.txt

2. **Test**: Reads the Senseval test file and uses Naïve Bayes reasoning to classify each test with a word sense. It outputs this data to *outputFiles\\output_[**word**].txt.*
3. **Analyze**: Performs an automated analysis on the test results by comparing the output data to the Senseval test key. The analysis is output to *outputFiles\\analysis_[**word**].txt.*

Driver

This program lets the user choose a word to disambiguate. Afterwards, it creates a Classifier object, which trains, tests and analyzes the particular word.

Analysis & Results

Background

Analysis was performed by running the program with three words: add, bank and expect. Default parameters were stated in the *Basic Setup* section. Much of the analysis was performed by accumulating, manipulating and reviewing data in Microsoft Excel.

Preliminary Results

Given our default parameters of window size +/- 4 and smoothing factor +1, the accuracy results (calculated as # of matches / # of tests) produced by the program were as follows:

Lexical Unit	Senses	Granularity[4]	Training Size	Test Size	Accuracy
add.v	6	Fine	263	132	**65%**
bank.n	10	Fine	262	132	**75%**
expect.v	3	Coarse	156	78	**74%**

Surprisingly, bank has the highest level of accuracy despite having the most senses. If we assume a random choice from classifications of equal probability, we have a 1/10 chance of classifying bank

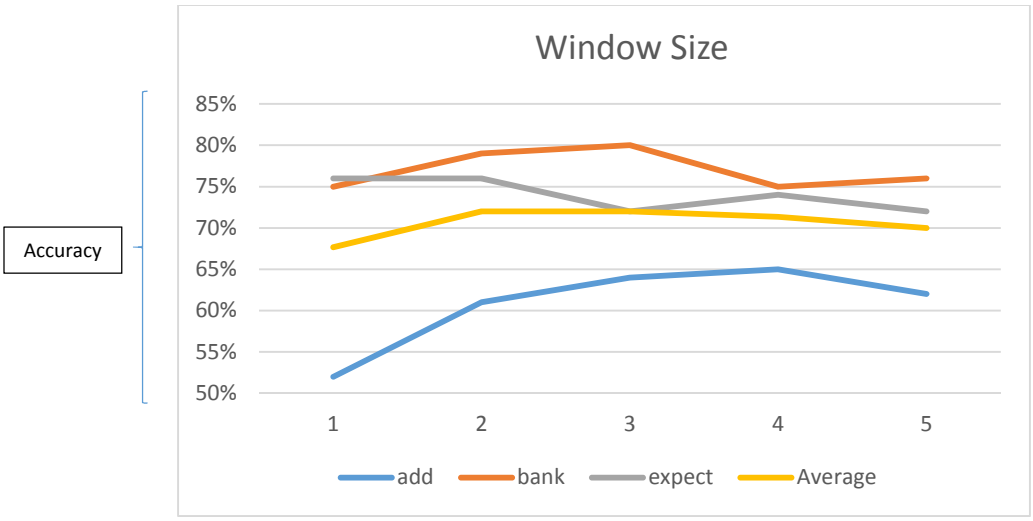
[4] Table 1 from http://acl.ldc.upenn.edu/W/W04/W04-0807.pdf

correctly and a 1/6 chance of classifying add correctly. Therefore, one might hypothesize that it should have the smallest accuracy. Nonetheless, that type of hypothesis fails to look at the contextual details of each word. If we consider a spectrum where one extreme is homonymy and the opposing extreme is polysemy, 'bank' lies more-so on the homonymy side than 'add'. By this, I mean bank has senses that are highly unrelated. This can be proven in *Exhibit 2A*, where the two most used senses of 'bank' are 1:14:00 and 1:17:01. Respectively, they mean "a financial institution" and "a sloping body of land near water". On the other hand, the top two senses for 'add' are 42601 ("to combine with something else") and 42603 ("to say beyond what was said"). Refer to *Exhibit 4A* for sense training distributions. As one might notice, there is more ambiguity to the top two 'add' senses than there are for the top two 'bank' senses. For bank, one can visualize concrete differences between its senses. Thus, we can more easily come up with more concrete discriminatory words to detect its class (water, money).

When we consider the good relative-performance of the term 'expect', we can consider its significantly small level of granularity (half as many senses as 'add'). Furthermore, we can see in *Exhibit 3A* that 99% of its uses relate to only two senses. Therefore, we are essential guessing between two classifications. This explains its relatively good performance. Why is it not performing better than bank? This could be for the same reason as 'add'. Additionally, the sample size is smaller, so it may be less representative.

Parameter Experimentation: Window Size

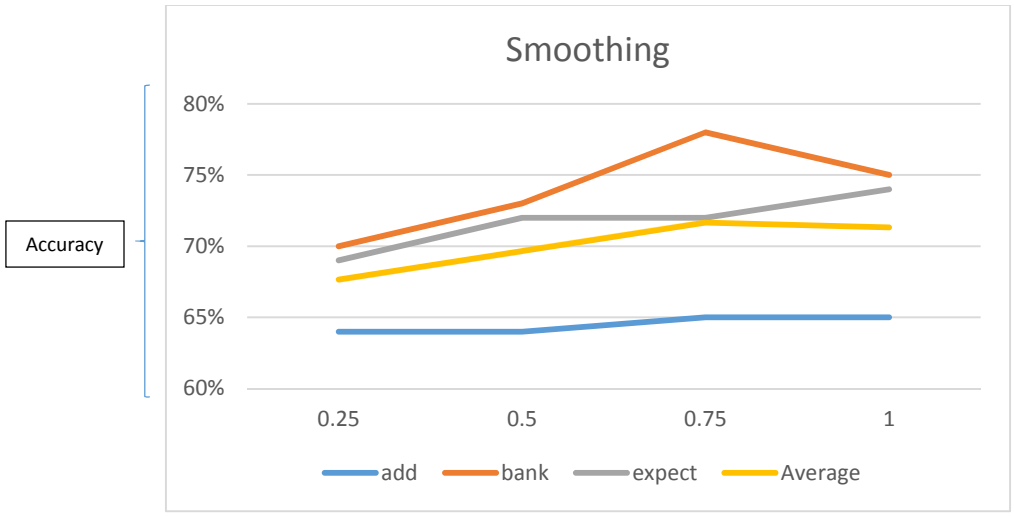
The following graph is based off of data in *Exhibit 1A* of the Appendix. A window size of 2 or 3 produced the highest average accuracies. This is mainly due to strong results from bank. Despite giving more context, a larger window size did not necessarily improve results (past 2 or 3). This is likely due to decaying value as we get further away from the target word.



Some might say that 4 or 5 words is not far enough to start decaying the value of context words. But to understand why this threshold is relatively small, we must consider the implementation of the Classifier class. When analyzing a training instance, we used an *adaptive* context window. Let us assume we have a context window of +1 and we are analyzing a sentence with target word w_n. If the word w_{n+1} is an invalid word (for example, a stop word) then we will consider w_{n+2} for context data. We continue this process until we reach the training sentence boundaries or until we have obtained an equivalent number of valid words that fits our context window size. This adaptive and expanding aspect of our context window means that although we are saying +/- k, we could easily be looking at words even further away than k (until we have 2k context words). Thus, the window is not necessarily as small as one might believe.

Parameter Experimentation: Smoothing Factor

The following graph is based off of data in *Exhibit 1B* of the Appendix. On average, results tended to be better around a smoothing factor of .75 to 1. One might hypothesize that a lower smoothing factor should have improved results since it does not alter the probabilistic data as significantly. Why did the higher smoothing factors perform better?



As we can see in *Exhibits 2A*, *3A* and *4A*, number of training instances per sense were not necessarily uniform. For example, 171, or 64% of the training instances for bank related to a single sense. Other senses spanned from less than 1% to 15% of total training instances (See *Exhibit 2A*).

Exhibits 2B, *3B* and *4B* present the results for 'bank', 'expect' and 'add,' respectively. Each cell (in each table) represents either a match or mismatch between Senseval key results and the program results. If we consider the Grand Totals of the program results, we can see that they were highly correlated with the distribution of the training sample. This is likely due to the fact that, with more training samples, a particular sense has more data (context words) and less susceptibility to sparseness. Therefore, it has a higher likelihood of a good score. Therefore, when considering the handicap due to sparseness for senses with low training frequencies, a relatively higher smoothing factor could mitigate this handicap, which justifies a smoothing factor of 0.75 to 1.

Conclusion

In conclusion, our experimentation proved to have initially counterintuitive results. Nonetheless, under further analysis of detail, we believe these results to be rather reasonable.

- Accuracy does not necessarily counter-correlate with granularity. This is because there is more to a word than its number of senses. One must also consider its nature (i.e. its preliminary level of ambiguity), as well as its training set.
- A larger context window size gives each sense more data. Nonetheless, it may not necessarily be valuable data. Given a target word in a sentence, context words may decay in value as we get further from the target word.
- Although we might expect a lower smoothing factor to be more representative of true data, this was not the case for our current samples. Factors around .75 to 1 appeared to give better results. This could be due to the uneven distribution of sense data. Some senses had only one or two training instances. Thus, they had very minimal probabilistic data, which leads to sparseness (many zero probabilities where other senses have high probabilities). Allocating a higher smooth factor to such handicapped senses strengthened them.

Areas of Improvement

Analysis of more words

Due to time constraints, only three words were considered: add, expect and bank. Improvements of analysis could be obtained by a full analysis of all words tested in Senseval-3. This would allow a more comprehensive exploration in potentially-varying results produced by testing words with different sample sizes, parts-of-speech and numbers of senses. We could even go so far as to look at other data sets (For example, Senseval-1 & Senseval-2).

Larger sample size per word/sense

As we saw in our analysis, some senses only had one or two training instances, which gave it a probabilistic disadvantage due to sparseness. Accuracies may be enhanced by using a larger and more representative sample size of training and test instances per word and per sense.

Training Sets & Cross-Validation

Given the input from Senseval, there is only one training set and one test set per word. With more time and effort, we could hypothetically create multiple training and tests sets using cross-validation. This might improve our results because we might find better fits between our training and test data.

Experimentation with other classification algorithms

Experimentation with other algorithms, such as decision trees and discourse properties could be performed. This could even include different implementations of the same algorithm. For example, WEKA contains a few different adaptations of Naïve Bayes. Afterwards, results could be compared between the various algorithms to analyze which ones were more accurate.

Program Execution Instructions

1. Unzip project file.
2. Open Eclipse.
3. Import *WordSenseClassifier* as existing project into workspace.
4. Run Driver class as Java Application. Program will commence in Console window.
5. Enter desired word to analyze (including '.n' or '.v'). Press Enter.
6. Once program has finished running, see related output file in outputFiles directory. (Open in Microsoft Excel for better column alignment and data manipulation.)

Appendix

Exhibit 1A: Varying Window Size

Given a constant smoothing factor of +1

	Window Size				
Word	**1**	**2**	**3**	**4**	**5**
add	52%	61%	64%	65%	62%
bank	75%	79%	80%	75%	76%
expect	76%	76%	72%	74%	72%
Average	68%	72%	72%	71%	70%

Exhibit 1B: Varying Smoothing Factor

Given a constant window size of 4

	Smoothing			
Word	**0.25**	**0.5**	**0.75**	**1**
add	64%	64%	65%	65%
bank	70%	73%	78%	75%
expect	69%	72%	72%	74%
Average	68%	70%	72%	71%

Exhibit 2A: Bank - Train Set Distribution

Distribution of senses via Senseval training key

Sense	Count	%
bank%1:04:00::	8	3%
bank%1:06:00::	26	10%
bank%1:14:00::	171	63%
bank%1:14:01::	7	3%
bank%1:17:00::	11	4%
bank%1:17:01::	42	15%
bank%1:17:02::	1	0%
bank%1:21:00::	2	1%
bank%1:21:01::	1	0%
U	4	1%
Total	273	100%

Legend for Exhibit 1B:

A: From Senseval Key

B: From Program Results

FALSE section: mismatches in key vs. results

TRUE section: matches in key vs. results

Total exceeds training size because some instances have 2+ classifications.

Exhibit 2B: Bank - Results

B

A

	bank%1:06:00::	bank%1:14:00::	bank%1:17:00::	bank%1:17:01::	Grand Total
FALSE	**4**	**15**		**14**	**33**
bank%1:06:00::		6		1	7
bank%1:14:00::	2			8	10
bank%1:14:01::	1	2		2	5
bank%1:17:00::		2			2
bank%1:17:01::	1	2			3
bank%1:17:02::		1		3	4
bank%1:21:00::		1			1
U		1			1
TRUE	**4**	**79**	**1**	**15**	**99**
bank%1:06:00::	4				4
bank%1:14:00::		79			79
bank%1:17:00::			1		1
bank%1:17:01::				15	15
Grand Total	**8**	**94**	**1**	**29**	**132**

Exhibit 3A: Expect - Train Set Distribution

Distribution of senses via Senseval training key

Sense	Count	%
1440301	116	66%
1440302	59	33%
1440303	1	1%
U	1	1%
Total	177	100%

Total exceeds training size because some instances have 2+ classifications.

Exhibit 3B: Expect - Results

Refer to Exhibit 1B for legend.

	1440301	1440302	Grand Total
FALSE	**14**	**6**	**20**
1440301		5	5
1440302	14		14
1440303		1	1
TRUE	**51**	**7**	**58**
1440301	51		51
1440302		7	7
Grand Total	**65**	**13**	**78**

Exhibit 4A: Add - Test Set Distribution

Distribution of senses via Senseval training key

Sense	Count	%
42601	138	46%
42602	1	0%
42603	88	29%
42604	6	2%
42605	8	3%
42606	58	19%
U	2	1%
Total	301	100%

Total exceeds training size because some instances have 2+ classifications.

Exhibit 4B: Add – Results

Refer to Exhibit 1B for legend.

	42601	42603	42605	42606	Grand Total
FALSE	**24**	**18**		**4**	**46**
42601		8		2	10
42603	16			1	17
42604	2	1		1	4
42605		2			2
42606	5	7			12
U	1				1
TRUE	**47**	**36**	**1**	**2**	**86**
42601	47				47
42603		36			36
42605			1		1
42606				2	2
Grand Total	**71**	**54**	**1**	**6**	**132**

References

Senseval-3 English Lexical Sample Task

http://www.senseval.org/senseval3

Stop Words

http://www.textfixer.com/resources/common-english-words.txt