Introduction:

Word Sense Disambiguation (WSD) is the problem and process of inferring the meaning of a word from its context. A simple example involves the word bank:

"You can deposit your monthly salary at the bank."
"A river bank is a common setting in neo-classical art."

Intuitively, the first sentence should refer to banks as a financial institution, while the second explicitly refers to a river bank. However, while this example is fairly straightforward, it is vastly more difficult to infer meaning from an arbitrary sentence. Two major difficulties generally arise: word sense definitions may themselves be ambiguous or only subtly different, and inferring the correct sense may require outside knowledge and/or semantic comprehension.[1]

This paper will begin with a brief overview of previous work on WSD with an emphasis on approaches and concepts. We will then proceed with a description of our goal, which is to produce a WSD system that can work without an explicit dictionary of word senses. We will then discuss our implementation, our method of evaluating the success of our system, and finally our results and conclusion.

Background:

In a survey on WSD algorithms, Xiaohua Zhou and Hyoil Han discuss two broad categories of algorithms and knowledge. The algorithmic categories are unsupervised and supervised, and they divide knowledge between lexical knowledge and learned world knowledge (Zhou and Han, 2005).

Lexical knowledge consists of sense frequency[2], sense glosses[3], concept trees[4], selectional restrictions[5], subject code[6], and part of speech. Learned world knowledge

---

[1] Zhou, Xiaohua, and Hyoil Han. "Survey of Word Sense Disambiguation Approaches." *FLAIRS Conference*. 2005.
[2] Literally the frequency with which a given word sense tends to appear in natural language.
[3] A sense gloss attempts to provide a brief explanation of a word sense.
[4] A concept tree attempts to "represent the related concepts of the target in the form of semantic networks as is done by WordNet" (Zhou and Han, 2005).
[5] "[T]he semantic restrictions placed on word sense… [f]or example, the first sense of *run* is usually constrained with human subject and an abstract thing as an object" (Zhou and Han, 2005).
[6] A subject code is an abstraction that attempts to categorize what a specific sense of the word belongs to. For example, we could categorize the most common sense of "run" to be "activity" or "sport", meaning that if we detect the topic of the sentence to be activity or sport, we may be more likely to choose this sense of "run".

consists of indicative words[7], syntactic features such as sentence structure and word presence, domain-specific knowledge, and parallel corpora, which attempts to use some third-party software to produce a translation of the original sentence to be used as additional information for interpreting sense.

Our primary interest moving forward lies in what Zhou and Han describe as supervised learning algorithms. Namely, an algorithm which is mostly expected to build up its knowledge base, rather than receiving it a priori. However, we will describe three of the unsupervised algorithms. Zhou and Han describe a Simple Algorithm (SA) which relies on one type of lexical knowledge[8], an Iterative Approach (IA) that only tags words with high confidence, and which uses past tagged words to attempt to tag future words, and a Bootstrapping approach (BS), which takes in a few seeds of training data and then continuously optimizes its internal model until convergence.

Of the supervised approaches we have: the Log Linear Model (LLM), which chooses the most probable sense with a structure analogous to the Naive Bayes approach for document classification; Decomposable Probabilistic Models (DPM), which attempts to avoid the independence assumption of LLM by constructing interdependence statistics from the training corpus; Memory-based Learning (MBL) which works to classify new cases by extrapolating from similar already-classified cases; and Maximum Entropy (ME), which builds a constrained optimization problem from the data that attempts to maximize the entropy of $P_\lambda(y \mid x)$, the conditional probability of a sense *y* given the features *x*.

Banerjee and Pedersen describe the Lesk Algorithm[9], which aims to "disambiguate[] words in short phrases" (Banerjee and Pedersen 2002, p. 18). Intuitively, for each word *w* in a sentence the algorithm looks at the adjacent words as a phrase, and assigns a sense for each word based on which senses share the most overlap. Banerjee and Pedersen present the below example for the phrase "pine cone".

Say *pine* has two senses:
Sense 1: kind of **evergreen tree** with needle-shaped leaves.
Sense 2: waste away through sorrow or illness.

---

[7] Words around the target word which provide context for the appropriate sense. Generally, closer words to the target are considered more indicative.
[8] Most commonly sense frequency, sense glosses, concept trees, selectional restrictions, and subject code. More interestingly, the authors also mention a simple benchmark algorithm that simply assumes the most popular sense of the ambiguous word.
[9] Banerjee, Satanjeev, and Ted Pedersen. "An adapted Lesk algorithm for word sense disambiguation using WordNet." *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer Berlin Heidelberg, 2002.

Say *cone* has three senses:
Sense 1: solid body which narrows to a point.
Sense 2: something of this shape whether solid or hollow.
Sense 3: fruit of certain **evergreen tree**.

The Lesk algorithm will notice that two words overlap in Sense 1 and Sense 3 respectively, and so assigns those senses to the two words. In order to evaluate their implementation of this algorithm, the authors used data provided by the SENSEVAL-2 competition.

Cohen, Schvaneveldt, and Widdows evaluated the effectiveness of Random Indexing (RI) relative to Latent Semantic Analysis (LSA) in their ability to infer "meaningful connections between terms that are related but do not occur together in any document in a collection" (Cohen, Schvaneveldt, Widdows 2010, p. 240). In particular, they look at the applicability of these algorithms "as [a] means to discover implicit connections in the biomedical literature without the need to explicitly identify a bridging term" (Cohen, Schvaneveldt, Widdows 2010, p. 241).

They note that Random indexing has "an enormous enormous computational advantage over methods requiring [singular value decomposition], in both space and time requirements" (Cohen, Schvaneveldt, Widdows 2010, p. 244). However, they also state that "LSA produces more interesting indirect neighbors" than several proposed implementations of RI, but also proposes a variant called "Reflective Random Indexing", which attempts to encourage similar vector representations across documents by a twofold process: building random vectors for terms, building a document vector based on those terms, then returning and rebuilding the term vectors based on the document vector (Cohen, Schvaneveldt, Widdows 2010, p. 246).

Goal:

A notable difficulty with many of the above papers is a dependence on predetermined word senses, either from a remote data source or manually inputted. Such data is difficult to tag objectively[10], and limits the applicability of software to domains which are expected to share word senses. Our goal in this paper is therefore to write a program that is able to determine the sense of an ambiguous word without explicitly providing possible senses for each word.

[10] "Even if trained linguists manually tag the word sense, the inter-agreement is not as high as would be expected." (Zhou and Han 2005, p. 1)

Implementation and Data:

Our implementation relies on random indexing to construct context vectors for each word. Specifically, for each word *w* we produce a lemmatized form *l(w)*. We then map that lemmatized word to a sparse random vector *v*. Every such vector is of length 2000, and contains 100 elements which are arbitrarily chosen to be 1 or -1.

The program then goes through the training corpus, building up context vectors for each word. Specifically, for every word in every sentence, it adds the index vector for a few words around it on either side to the middle word's context vector. We cumulatively build up this context vector for all sentences in our training corpus. This only has to be done once

Once we have these context vectors, we accept a sentence $s = w_1 w_2 w_3 ... w_n$, with a specified ambiguous word w_i. We also produce a list of all sentences from our training corpus which also include w_i. We then try to deduce the sense of the ambiguous word by returning sentences from our training corpus which we believe uses w_i in the same sense. After retrieving all sentences that use the given word in the training corpus (stored in the concordance), we compute a score for similarity between the input sentence and each training sentence match with respect to the given ambiguous word.

Two scoring methods were tested: Sentence Match Score (SMS), similar to the Lesk algorithm, compares the context vectors of words in a window around the ambiguous word in both the input and training sentence, and Word by Word Score (WBW), which compares individual word context vectors between the two sentences. Our final program uses SMS scoring, which was easier to implement and seemed to yield better results. We finally sort the list of matches by this newly computed score and return the top results.

We also implemented an additional feature: namely returning a ranked list of definitions for the target word from WordNet rather than sentences from the training corpus. In this case, we always score by constructing a context vector that represents the training sentence and the input sentence, and calculating the cosine similarity of those vectors. Our initial purpose for this feature was to serve as an evaluation metric, but is currently beyond the scope of this project.

We rely on two datasets: a subset of the Guardian corpus, as well as a much smaller test file containing sentences which use the word “bank”. Both data sets are provided with the code. Libraries used include Apache’s OpenNLP toolkit, FST fast serialization for data persistence, and JWNL to access WordNet for dictionary definitions.

Testing/Evaluation:

Quantitatively evaluating the results of our form of word sense disambiguation is extremely difficult because we are not retrieving a specific definition but rather a list of results where order matters. One way we could turn our task into a more concrete machine learning problem would be to use a dataset with lots of text where every single word is tagged with its true usage/definition. Creating this dataset manually would be monumentally tedious for any significantly sized training corpus, or would possibly require some crowdsourced data-mining method (such as Amazon MTurk). If it did exist, we could then simply use any canonical machine learning algorithm such as k-nearest neighbors and its associated evaluation methods. In the end, however, we do believe that our program’s output of a ranked list of example sentences is more useful than a program which would only output a dictionary definition.

One reason it’s more useful is flexibility. We can, for example, use dictionary definitions as our “example” sentences; meaning our program in effect then ranks all definitions for an ambiguous word by which one matches the input sentence in context the closest. We can then just create a test corpus with test inputs and the true dictionary definition of each ambiguous word’s usage, and compare them with which definitions rank highest in our program’s output for each input.

Results:

Our results were promising, although we do not currently have a quantitative metric for measuring it. Unfortunately, our program’s performance was not consistent over all words and sentences, performing very well in some cases, but not so much in others. We leave a single example below to illustrate the program.

Please enter a sentence.
The river bank has a lot of sand.
Which word in that sentence is ambiguous?
bank

# of results: 163

0.4638341070007304 pinterest <http ://www.pinterest.com/pin/create/button/ ?description=italian+village+trattorias %3a+readers %e2 %80 %99+travel+tips&url=http %3a %2f %2fgu.com %2fp %2f4km2e %23img-5&media=https %3a %2f %2fmedia.guim.co.uk %2f8533e7578d9bc7c03e1ecea227fb2bd99292f154 % 2f0_129_960_576 % 2f960.jpg> this is a cosy , family-run trattoria on the banks of the arno river , in a valley north of florence .

0.4593896231064596 however , you could take a train to ivybridge then a taxi for the three-mile journey to the start , or pick up thetwo moors way national trail <http ://www.devon.gov.uk/walking/two_moors_way .html> , which starts in the town – and join this route at step 2. top tip if you want to walk out by a different route from the one by which you arrived , you can continue south up to stall moor and pick up the path that runs alongside the west bank of the river erme . it will take you back to the road that leads to harford . the downside is a 1.5-mile road walk back to harford . the plan the dartmoor route day 1 1 . from the car park head onto the open moorland in an easterly direction .

0.44857882238840224 built beside an ancient pilgrimage route on the banks of the river ganges , 5km upriver from rishikesh's famous laxman jhula suspension bridge , the ashram taps into the holy town 's spiritual vitality while eschewing its chaotic hustle .

0.4410607990059353 the guides and piste bashers , who had also pulled up a tent for the craic , had of course banked a perfect night's sleep.

0.4366707768420968 "where the state has had to bail out a bank , the treatment of its business customers is a matter of the highest public importance , if that treatment has implications for the exchequer and taxpayer , "he wrote .

0.42607094726458683 on monday , the irish times said the attempt to use the earlier court injunction to gag reporting of parliament was "an attack on the public interest right to know the details of the banking arrangements of one of the country 's richest men" . in an editorial <http ://www.irishtimes.com/opinion/editorial/in-defence-of-parliament-1.2232589> , it described the legal move as "a deeply worrying erosion of a pillar of our parliamentary system" .

0.4250743904921626 all this in an eminently green city , laced with 750 miles of tree-lined bike paths and the winding banks of the isar river – perfect for picnicking or swimming .

0.4214860162118159 set on the banks of the douro in the north of the country , porto's historic centre has been unesco-listed since 1996 and is a picturesque mish-mash of medieval churches , cobbled lanes , pretty squares , steep steps and beautiful buildings tumbling down to the river .

0.42009560888673403 one minute they were all laughing it up at a christening on the banks of the river jordan ; now tony's probably having to send little grace and chloe their presents via the pentagon .

0.40387625188531134 also on the north bank of the brisbane river , once-seedy caxton street , near paddington , is a new social hub , now home to the flamboyantgambaro hotel <http ://www.gambarohotel.com .au/ > and to lefty's oldtime music hall <http ://leftysmusichall.com/> ,

which has live bands , 100 rye whiskies , and was named gourmet traveller's "best australian bar 2014" .

Conclusions:

Our approach to and implementation of word sense disambiguation demonstrates some promising ideas; instead of using traditional machine learning methods, we believe our program's output of a "ranked retrieval" yielded some positive results, but still requires much tweaking to find what specific scoring methods and constants work well for this approach. Using dictionary definitions at retrieval time, an idea which we only implemented towards the end of our project, turned out to be a relatively successful feature as well, hinting that we could possibly treat this task as a classification problem given more time. This would also allow a more concrete quantitative evaluation metric for our results, something which the "ranked retrieval" method was lacking and which made it hard to test and evaluate our program's performance.

References to Literature:

Banerjee, Satanjeev, and Ted Pedersen. "An adapted Lesk algorithm for word sense disambiguation using WordNet." *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer Berlin Heidelberg, 2002.

Cohen, Trevor, Roger Schvaneveldt, and Dominic Widdows. "Reflective Random Indexing and indirect inference: A scalable method for discovery of implicit connections." *Journal of biomedical informatics* 43.2 (2010): 240-256.

Zhou, Xiaohua, and Hyoil Han. "Survey of Word Sense Disambiguation Approaches." *FLAIRS Conference*. 2005.