A Genetic Algorithm using Semantic Relations for Word Sense Disambiguation

Michael Hausman

Approved By

____________________
Advisor: Jugal Kalita

____________________
Committee member: Edward Chow

____________________
Committee member: Al Brouillette
Introduction

Word sense disambiguation is another way of saying, “which dictionary definition is correct in this context?” One possible way to answer this question is to take 100 different linguists and ask all of them to tag each word with the appropriate definition. This would immediately show that some definitions are too ambiguous for linguists to agree on the definition, especially on text designed to have two meanings. While it would be entertaining to watch the linguists argue over every word, this is tedious and impractical to do on every piece of writing. The goal is to have a computer find the appropriate definitions to make the process much faster.

There are several approaches for a computer to guess at the definitions, such as path based methods and gloss based methods. Path based methods look at frequency statistics and other word semantic measures to measure which definition is correct. These methods typically use tools, such as WordNet, to get the measures. Gloss based methods take a definition and finds how many words from the definition are in the surrounding text. The idea is that words in a specific context will be similar. This project takes several path based measures to calculate the word senses in a genetic algorithm.

Genetic Algorithm Overview

Word Sense Disambiguation for all words in a file creates an incredibly large solution space. For example, there exists a file can with 5 paragraphs of information. Each of these paragraphs could have 100 words, which means the file has 500 words. Each word can have many definitions, so assume there are 5 definitions per word. This makes $5^{100} = 7.888 \times 10^{69}$ possible word sense combinations per paragraph or $5^{500} = 3.055 \times 10^{349}$ word sense combinations for the entire file. This solution space is far too large to check every word sense combination. If there is a way to measure the accuracy of a word sense combination, genetic algorithms is a way to check a subset of these solutions for a good solution. This solution will not be “the solution,” but the solution should be close.

The genetic algorithm is based on Darwin’s theory of evolution to solve optimization problems. The general idea is to start with a set of solutions, use that set of solutions to make better solutions, and only keep the best solutions. In genetic algorithms, “chromosomes” define all of the information necessary to define a solution. The individual pieces of the chromosome are “genes” that define a certain aspect of the solution. In this case, the chromosome defines the word senses for each paragraph. Each gene represents a word sense for a single word. The generic process of creating “better” chromosomes, or solutions, with genetic algorithms is below.

1. Start with a set of solutions (1st Generation)
2. Take original “parent” solutions and combine them with each other to create a new set of “child” solutions (Mating)
3. Somehow measure the solutions (Cost Function) and only keep the “better” half of the solutions
   - Some may be from “parent” set, others are from “child” set
4. Introduce some random changes in case solutions are “stuck” or are all the same (Mutation)
5. Repeat starting with Step 2

Chromosome

The chromosome for this project is similar to the one developed by zhang et al. [1]. The chromosome represents a paragraph of words. Each gene of the chromosome represents a word. The
value in each gene is a number representing the current sense the chromosome is using. The genes will
never switch positions and the chromosome will always be the same length. However, the genetic
algorithm will need to keep track of the part of speech for every word. A part of speech tagger that is
over 95% accurate will tag every word [2]. This part of speech will not change throughout the genetic
algorithm.

Cost Function

The cost function is perhaps the most important part of a genetic algorithm. Without a cost
function, there is no way to tell how good an answer is. If the cost function calculates the best answer
incorrectly or emphasizes the wrong solution, then the genetic algorithm is useless. One of the main
goals of this project is to obtain an accurate result, so the cost function must be able to separate the
best solutions from the inaccurate solutions. Therefore, the author spends great deal of time and effort
simply developing two possible cost functions.

In general, the cost functions rely on several semantic relations from Princeton University’s
WordNet program. WordNet is a machine readable dictionary that also reports several relations
between words [3]. Calculating the individual semantic relations and adding the results is one possible
cost function. Some relations, however, are more useful in some situations. Weighing these relations
for those situations is the second possible cost function. This project uses and compares the two cost
functions.

Semantic Relations

All the semantic relations come from WordNet. Each semantic relation uses some aspect of the
word to define how closely two words are related. The aspects and how to calculate them from
WordNet are below.

1. **Frequency**: WordNet displays all definitions of a word in the order it appears most often in the
corpus Princeton used to develop WordNet.

   \[ Freq(w) = \frac{SenseTotal(w) - Sense(w) - 1}{SenseTotal(w)} \]

   SenseTotal(w): The total number of senses for this word
   Sense(w): the sense number of the word currently in use

2. **Hypernym**: A hypernym of a word is a more generic way of saying that word. Having several
   hypernyms in a row creates a tree where words have a given rank. The equation below from
   Wu-Palmer and takes advantage of hypernym trees [1] [9].

   \[ SimWP(w_1, w_2) = \frac{2 \times depth(lso(w_1,w_2))}{(len(w_1, lso(w_1,w_2)) + len(w_2, lso(w_1, w_2)) + 2 \times depth(lso(w_1,w_2))} \]

   w1 and w2: the two words being compared
   lso(w1,w2): lowest super-ordinate, which is the most specific hypernym between the
two words
   len(w*, lso(w1, w2)): how deep to the lso from the word
   depth(lso(w1,w2)): how deep to the lso from the root of the hypernym tree
3. **Domain**: A domain is the field or collection a word belongs to.

\[
\text{Dom}(w_1, w_2) = \begin{cases} 
1, & \text{Dom}(w_1) = \text{Dom}(w_2) \\
0, & \text{Otherwise}
\end{cases}
\]

w1 and w2: the two words being compared
Dom(w\*): the domain of the word given

4. **Synonym**: Two words that can be interchanged in a given context are synonyms.

\[
\text{Syn}(w_1, w_2) = \begin{cases} 
1, & \text{Syn}(w_1) \supseteq \text{Syn}(w_2) \\
0, & \text{Otherwise}
\end{cases}
\]

w1 and w2: the two words being compared
Syn(w\*): all of the synonyms of the word
Syn(w1) \supseteq Syn(w2): if any of the synonyms from word 1 is a synonym of word 2

5. **Antonym**: Two words that are opposites of each other in a given context are antonyms.

Antonyms are calculated in the same manner as synonyms.

6. **Coordinate Sisters**: Two words that are the same rank in a hypernym tree are coordinate sisters.

These words should be closely related to the given word. Coordinate sisters are calculated in the same manner as synonyms.

**Semantic Relation Addition Cost Function**

Many earlier researchers focused on a single semantic relation to calculate how related two words are. Many of these authors built on each other’s work to find better ways of utilizing a single semantic relation. The hypernym calculation above is one example. Many later researchers found that combining several semantic relations has better results. Therefore, the first cost function takes several semantic relations into account. Simply separate a chromosome into every possible word pair combination. Then add all the relations above for each word pair combination as shown below:

\[
\text{Cost}(c) = \sum_{i,j} \left[ \text{Freq}(w_i) + \text{SimWP}(w_i, w_j) + \text{Dom}(w_i, w_j) + \text{Syn}(w_i, w_j) + \text{Ant}(w_i, w_j) \right]
\]

\[+ \text{Coor}(w_i, w_j), \quad i \neq j \]

c: the current chromosome
wi and wj: the current word pair combination

**Weighted Semantic Relation Addition Cost Function**

As Degemmis et al. points out, some semantic relations are better for certain parts of speech [4]. This makes sense because some semantic relations are not available for all the parts of speech. Adjectives, for example, don’t have a hypernym tree. Also, some semantic relations only work up to a point. Only using the most frequent definition will only be 100% correct when there is only one definition. Some of the other relations are better at finding the lower level definitions, but they cannot find all definitions. The semantic relation calculations often conflict because each semantic relation has different strengths in different situations. Therefore, there is theoretically an optimum mixture of the semantic relations that would minimize the impact of the conflicting calculations. To find this mixture for every part of speech, the author does the following.
1. Take a translated file and separate the parts of speech.
2. For each part of speech, find every possible word sense combination. Each word sense combination will be a chromosome. For example, there are two nouns, \( n_1 \) with 10 senses and \( n_2 \) with 3 senses. Then there are 30 possible chromosomes between these two nouns.
3. Calculate the semantic relations for every chromosome from step 2. For example, add up the calculated frequencies of a chromosome. This chromosome is one of the 30 noun possibilities from earlier.
4. For each semantic relation from step 2, make a scatter plot of a part of speech with every point representing a chromosome. The x axis is the ratio of correct senses. The y axis is the semantic relation calculation.
5. Make a trend line for each scatter plot. This trend line should have the best \( R^2 \) value for the data. However, do not use a trend line that goes below zero.
6. Use these trend lines to calculate a system of equations for a weighted version of the first cost function. Note that there should be a different system of equations for each part of speech. For example, \( 100 = \text{freq}_{\text{noun}}(100) \times \text{X} + \text{SimWP}_{\text{noun}}(100) \times \text{Y} \ldots \text{Coor}_{\text{noun}}(100) \times \text{Z}, \ 95 = \text{freq}_{\text{noun}}(95) \times \text{X} + \text{SimWP}_{\text{noun}}(95) \times \text{Y} \ldots \text{Coor}_{\text{noun}}(95) \times \text{Z}, \ldots \) etc.
7. This should calculate the weights for the new cost function.

\[
\text{Cost}(c) = \sum_{i \neq j} ^{\text{POS}_{i,j}} \text{Freq}(wi) \times \text{FreqWeight}(\text{POS}) + \text{SimWP}(wi,wj) \times \text{SimWeight}(\text{POS}) \\
+ \text{Dom}(wi,wj) \times \text{DomWeight}(\text{POS}) + \text{Syn}(wi,wj) \times \text{SynWeight}(\text{POS}) \\
+ \text{Ant}(wi,wj) \times \text{AntWeight}(\text{POS}) + \text{Coor}(wi,wj) \times \text{CoorWeight}(\text{POS}), \quad i \neq j
\]

c: the current chromosome  
wi and wj: the current word pair combination  
POS: the current part of speech  
***Weight(POS): the weight for the relation and part of speech calculated in step 6

The weights of the new cost function represent the optimum mixture for that part of speech. Once these weights are found on some training data, then there is no need to calculate them again. This weighted cost function is simply another way to estimate the accuracy of a result in a genetic algorithm.

**Mating**

The mating function will be a dominant gene algorithm [5] [6]. As said before, each gene represents a word sense. The cost function needs to keep track of the cost of each gene. This way the genetic algorithm can compare the genes of a chromosome as well. When mating, simply keep the best genes (dominant genes) from one parent based on the gene cost. All the other genes will come from the second parent. This will make a child gene as strong as the first parent while giving the opportunity to become stronger from the second parent’s genes.

**Mutations**

The mating function focuses on the dominant genes. Therefore, the mutation function focuses on the lower cost genes. In this case, mutations will do two main mutations. The first is to randomly change one word sense of the lower cost genes. The second is to pick a random semantic relation and calculate the best sense using that relation on a lower cost gene. This way the mutation function provides an intelligent guess for improving low cost genes.
Testing and Deliverables

In order to test the genetic algorithm, some previously disambiguated data is necessary. This project will use SemCor files for all training and testing [7]. SemCor is a common resource used by many papers on word sense disambiguation. With this common source, the author can make accurate comparisons to previous work. Also, this prevents any human error or confusion about ambiguous definitions. Many of the papers found that use SemCor test the results on several if not all the files in SemCor. Also, there is another UCCS student who is solving word sense disambiguation with the page rank algorithm. The results of this project will be compared to Michael Billot’s results from his implementation. Deliverables will include any code, files, binaries, and results from the project.

Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 15, 2010</td>
<td>Wrote any code relating to the cost functions or cost function calculations</td>
</tr>
<tr>
<td>November 30, 2010</td>
<td>Calculated the weights for the Weighted Semantic Relation Addition Cost Function</td>
</tr>
<tr>
<td>December 30, 2010</td>
<td>Wrote code for the mating, mutation, and genetic algorithm portions</td>
</tr>
<tr>
<td>January 15, 2011</td>
<td>Tested all code and have results</td>
</tr>
<tr>
<td>January 30, 2011</td>
<td>Contact Michael Billot to get his latest results</td>
</tr>
<tr>
<td>February ??, 2011</td>
<td>Paper written and submitted to advisors</td>
</tr>
<tr>
<td>February ??, 2011</td>
<td>Final presentation</td>
</tr>
</tbody>
</table>

References
