CS580 - Query Processing

Build Support Word and Dispute Word Dictionary

Simant Purohit
Akshay Thirkateh
Prasad Nair

INDEX

1. Overview
2. Problem Statement
3. Algorithm with Example
4. Implementation
1. Overview
The project aims to create a database of words that are classified as support words or dispute words from a very small set/seed of support words and dispute words. This process is done by extracting words from wordnet dictionary and running an algorithm on the data to create the dictionary.

Given a statement, we identify words in the statement that are suspected to be untrue.

For Example: Obama is a Christian

Here the word ‘Christian’ is suspected to be untrue. We need to find whether the statement is true or false based on the meaning of this word in this statement. If the statement is untrue then we need to remove that word and use another word, which is true, of the same data type.

2. Problem Statement
We give a query, for example “Obama is a XYZ”. This query returns a list of documents like d1, d2, d3… dn. Each document will have statements that may contain words that either support the hypothesis or reject it.

For Example: The query is … Sears tower is in the East.

This query may return many documents. The documents contain statements which are closely related to this.

Document 1: “I believe that Sears tower is in the East.”

Here the word “believe” supports the statement that “Sears tower is in the East.” Hence document 1 supports the query or statement.

Document 2: “I doubt that Sears tower is in the East.”

Here the word “doubt” does not support the statement that “Sears tower is in the East.” Hence document 2 opposes the query or statement.

Support words confirm the query or hypothesis and dispute words are those that disagree with the statement or hypothesis.

The problem requires us to produce a list of support words for a given seed word. We need to produce a list for each parts of speech of the seed word if that parts of speech can be derived from the input seed word. We repeat this process for the seed dispute word

3. Algorithm

Step1:
1. Initialize the seed_word.
2. Identify and initialize 4 candidate seed words for the 4 parts of speech of the seed_word. This is done by querying using the getDerivationallyRelatedForms(seed_word) function of the Wordnet API to obtain the parts of speech words.
3. If a word does not have a particular part of speech the list containing the similar words is left empty.
Step 2:
1. We add the candidate seed word to the potential support dictionary with its iteration number ‘L’ = 0. To generate candidate support words we iterate from 0 to K, where K is a predefined number of iterations. So here L is from 0 to K.
2. We insert each candidate seed word into Wordnet to get a list of candidate support words. This is done by querying using `getSynsets(seed_word,type)` to get synsets.
3. We add the words of these synsets to the potential support dictionary along with the iteration number ‘L’. Seed word has iteration level 0. After it is inserted into Wordnet and its synsets are obtained we increase the iteration by 1. Words in its synset are stored along with this iteration number.
4. We now input these support words in Wordnet to obtain a set of synsets and extract potential support words. If support word is not present in the dictionary then we add it and increase the iteration by 1. We do this till say, L = K.
   For example: Believe, Iteration 0. We increase iteration by 1. We put believe in wordnet to get Think, Consider, Conceive, Trust in Iteration 1. We increase iteration by 1. We put think in wordnet to get Suppose,… in Iteration level 2.

Step 3:
1. If `seed_word ∈ Synset S_i` and word `W ∈ Synset S_i` i.e. they are in the same synset, and there are ‘n’ such synsets, then
   ```
   \{ sim(seed_word,W) = 0;
   For Synsets i = 1 to n
   \quad relativeFrequency_seedWord = relative frequency of a synset S_i w.r.t. seed_word;
   \quad relativeFrequency_candSuppWord = relative frequency of a synset S_i w.r.t. candidate support word W;
   \quad sim(seed_word,W) = sim(seed_word,W) + (relativeFrequency_seedWord * relativeFrequency_candSuppWord);
   \}
   ```
2. If `seed_word ∈ Synset S_1` and word `W ∈ Synset S_n` i.e. they are in different synsets, and there is a sequence of words `W_1, … W_n` in synsets `S_1, ..., S_n` respectively such as `W_i ∈ Synsets S_i` and `S_{i+1}` i.e `W_i` has the meaning of synset `S_i` and synset `S_{i+1}`. Where `1 <= i <= n` Then,
   ```
   \{ sim(seed_word,W_1) = (relativeFrequency_seedWord * relativeFrequency_W_1);
   sim(seed_word,W) = sim(seed_word,W_1)
   For Synsets i = 2 to n
   \quad sim(W_i, W_{i+1}) = (relativeFrequency_W_i * relativeFrequency_W_{i+1});
   \quad sim(seed_word,W) = sim(seed_word,W) * sim(W_i, W_{i+1});
   \}
   ```
3. If `seed_word ∈ Synset S_1` and word `W ∈ Synset S_n` i.e. they are in different synsets, and there is a sequence of words `W_1, … W_n` in synsets `S_1, ..., S_n` respectively such that the
relationship between $W_i$ and $W_{i+1}$, where $1 \leq i \leq n$, is exactly one of either hyponym, hypernym, troponym, similar to or entailment. Then,

\[
\{ \\
\text{relativeFrequency}_W1 = \text{relative frequency of } S1 \text{ w.r.t. seed_word} \\
\text{relativeFrequency}_Wn = \text{relative frequency of } S_n \text{ w.r.t. candidate support word } W \\
\text{If there are } 'n' \text{ number of paths between seed_word and } W \text{ such that } S_i \text{ has an edge to } S_{i+1} \text{ and } S_{i+1} \text{ has an edge to } S_{i+2} \ldots, S_n \text{ The edges represent a relation such as hyponym, hypernym, troponym, etc.} \\
\}
\]

For $k = 1$ to $n$

\[
\{ \\
\text{sim(seed_word}_k, W_k) = \text{relativeFrequency}_\text{seed_word}_k * (1/((\text{pathlength}_k)^*\text{pathlength}+1)) * \text{relativeFrequency}_Wk; \\
\text{sim[k]} = \text{sim(seed_word}_k, W_k); \\
\text{synsetSet} \leftarrow [\text{Set of synsets in the path}] \\
\}
\]

For $k = 1$ to $n$

\[
\{ \\
\text{If (addssynsetSetk(synsetSetk) \_ returns False) // Synset already present in a previous path} \\
\text{formulaToBeUsed = 1} \quad // \text{Dependent paths} \\
\text{else} \\
\text{formulaToBeUsed = 2} \quad // \text{Independent paths} \\
\}
\]

If (formulaToBeUsed == 1)

\[
\text{sim(seed_word,W) = max of all (sim[k]);} \\
\text{else if (formulaToBeUsed == 2)} \\
\text{sim(seed_word,W) = sum of all (sim[k]);} \\
\}
\]

4. Repeat all the steps for the dispute seed word to obtain potential candidate dispute words.

5. For Polarity validation: // Not implemented

   a. If candidate support word begins with “un”, “dis”, “anti”, “mal”, “non” then it is likely that it may be wrong as the support word has a negative polarity. If a dispute word $W$ contains these prefixes then it could be made into a potential candidate support word if it exists by removing the prefixes and then calculating its similarity.

\[
\{ \\
\text{If ("dis" or "mal" or "anti" or "un" or "non" ) is a prefix of dispute word } W \\n\text{Then} \\
\text{candidateSupportWord = (W – prefix)} \\
\text{if (candidateSupportWord Not in potentialSupportDictionary)} \\
\text{(calculateSimilarity(seedSupportWord,} \\
\text{candidateSupportWord)} \\
\}
\]
if sim(seedSupportWord, candidateSupportWord) > 0
add candidateSupportWord to potentialSupportDictionary
}

b.
i. Compare words in the potential support word list to those in the potential dispute word list.
ii. Identify words that are common in both the lists.
iii. Compare the similarity values of the word and keep the word with the higher similarity in its respective potential dictionary and remove the word from the dictionary which has a lower similarity.

6. The support word dictionary created without using a seed word may not be a valid support word for the given seed_word. For example:
a. For a seed_word ‘believe’, we get a derivationally related adjective, believable. Believable is a stemmed adjective of its related adverb ‘believably’.
b. A potential adverb support word for believably i.e. ‘credibly’ has a very low similarity to the seed word (because it has a relativeFrequency of 1 and has a low polysemy count = 2 i.e. the number of synsets that contain the word is 2, hence used rarely A very low relativeFrequency with respect to the seed word yields a very low similarity. As the similarity between the seed word and this derivationally related form is very low it may not convey the same meaning, thus it may not support the hypothesis satisfactorily).
Thus, a support word dictionary created without using a seed word may not be that effective. // But we have used support words to get adverb forms of the support words. We then use these words as seed words to build the adverb dictionary.

5. Implementation

The program would be written in Java, also the Wordnet API. The API is available for download from Princeton’s website. We would store the data and the results in the database. The input to the program would be the seed word which is doubtful. The output for the program would be a set of support and dispute words for each part of speech along with their similarities in the descending order, with the seed word in that part of speech.

Tools Used:
Eclipse- juno
MySQL Workbench

API used:
Wordnet - JAWS

Timeline:
Proposal: 10th Oct
Second Phase: 11st Nov ➔ Resubmit Proposal. Implement verified algorithms
Final Phase: 20th Nov ➔ Merge work
Testing: 25th Nov ➔ Test for errors
Submission: 26th Nov ➔ Final Report Submission