CS582 - Information Retrieval

Construction of Support and Dispute word Dictionary

Rajeev Reddy Vishaka ([rvisha2@uic.edu](mailto:rvisha2@uic.edu))

**OBJECTIVE:**

To construct a dictionary of Support words and a dictionary of dispute words from few seed support words and seed dispute words by extracting the words automatically from word net dictionary.

**OVERVIEW**

Given a doubtful statement with the doubtful unit D, find if the doubtful statement is true if it is not, then find the truthful one.

E.g.: Obama is a Muslim.

The above statement is doubtful.

Doubtful unit: Muslim.

We have to find the truthfulness of the statement. In order to do so, we have to replace the doubtful unit with the truthful one. This can be done by replacing the doubtful unit by truthful unit which has the same data type as doubtful unit.

It is a multi-step process

1. Find another statement by replacing doubtful unit by alternative unit.
2. Rank the alternative unit.

Step 1

* + Submit the query to the web. Query is doubtful statement-doubtful unit.

E.g.: Obama is a Muslim

* + From the retrieved result records, extract terms to possibly replace doubtful unit.
  + Each candidate term is related by the criteria that, it must have the same data type as that of the doubtful unit.

E.g.: Christian, which is an alternative unit to Muslim, has the same data type as that of Muslim. Both come under the category of religion.

Step 2

* Retain only those alternative unit if they fit the data type of the doubtful unit and they are highly correlated with the query.

Step 3

* Rate the alternative unit using no of hits and reliable sources (e.g., Wikipedia).

Our project is to construct the dictionary of support words and dispute words which can be used to determine the truthfulness of the statement.

We have to construct the dictionary of support words and dispute words from the seed input words. We are considering the approach of populating support and dispute words by specifying rules that picks apt words from WordNet. Given the nature of the problem and the steps taken, the project consists of three major parts:

1. Problem analysis
2. Algorithm
3. Implementation and Results

**Dictionary Used**

***Word Net***

Word Net is a free word database provided by Princeton University. It can be downloaded for free from cognitive science laboratory website of Princeton University. It serves as dictionary as well as thesaurus.

1. **Problem Description and Analysis**

If d1, d2, d3…... dn are the documents retrieved (SRR’s) after giving the query (e.g. Obama is a), each statement will have alternative unit but we are not sure whether it is supportive or not.

For e.g.: Let q: “John is a good man” be the query given.

* Document d1 contains “I believe that John is a good man”.

Here believe supports the statement “John is a good man”. Hence, Document d1 is supporting the statement.

* Document d2 contains “I doubt that John is a good man”.

Here doubt does not support the statement “John is a good man”. Document d2 is therefore opposing the statement.

Support words are those which have an affirmative opinion about the statement and dispute words are those which have negative opinion about the statement.

1. **Algorithm**

The Algorithm is as follows:

1. The relative frequency of a synset s with respect to a word w is the frequency of s in w divided by the sum of frequencies of the synsets of w. This is the probability that w has the meaning of synset s.
2. Similarity(word w in synset s, synset s) = probability that word w has the same meaning of s=

relative frequency of synset s with respect to word w.

1. Similarity (word w1 in synset s1, word w2 in the same synset s1) = probability that w1 has the meaning of synset s1 \* probability that word w2 has the meaning of synset s1.
2. Suppose that there are multiple synsets containing both word w1 and word w2. Let the synsets be s1, s2, ..sk.

Similarity(w1, w2) = sum(j from 1 to k) similarity(w1 in synset sj, w2 in synset sj).

1. Suppose no single synset contains both words w1 and w2 but there is a synset s1 containing w1, another synset s2 containing w2 and there is a path between synset s1 and synset s2 via their least common ancestor.

Similarity(w1, w2)= similarity(word w1 in synset s1, synset s1) \* 1/(length of the shortest path between s1 and s2 via their least common ancestor +1) \* similarity(word w2 in synset s2, synset s2). If there are multiple synsets s11’, s12’.. s1n’, each containing w1 and there are multiple synsets s21’, .. s2n’, each containing w2 and there is a path between S1j’ and S2j’, for each j from 1 to n’, then similarity(w1, w2) = sum (j from 1 to n’) (similarity (word w1 in synset s1j’, synset s1j’) \* 1/(length of the shortest path between s1j’ and s2j’ via their least common ancestor +1) \* similarity(word w2 in synset s2j’, synset s2j’).

1. This idea applies to finding similarity between believe and a word in a hypernym synset of believe. The same principle applies to troponyms .
2. Rank the list of words in descending order of similarities with “believe”.
3. For each part of speech of “believe”, obtain such a list.
4. Do the same for “dispute”.
5. First the words are processed in to Database and each time a word is added to dictionary a sort is performed on similarity to take the next word with highest similarity.

Example : believe is taken, and its synsets, hypernyms, hyponyms, derivationally related words, troponyms are obtained. Then a sort is made and next hghest similarity word is taken. We now relate all the forms of this word with seed word believe.

1. Like that we perform recursively. For every 101 words generated we move in to dictionary just to save time(open and close connection for every word). A huge word set is generated and we stop this algorithm when we continuously get 20 to 30 words with similarity zero. If there is no stopping condition this algorithm keeps on generating words and such similarity of those words is found to be zero at the last.
2. Similarity will be only found between words of same sense.

This algorithm is efficient to even covers sim(belief, believer) which are both derivationally related words of believe and noun sense. Seed word becomes belief, for word believer and seed word becomes believer for belief.

**Example**

1. **For step1 in algorithm:**

If the word w=believe, synset s=believe

The relative frequency of synset(believe) w.r.t to word believe is =119/(119+72+37+10+2)=119/240.

The relative frequency of synset is defined as the frequency of synset w.r.t word to the sum of the frequencies of the synsets of word w.

1. **Step 2 in algorithm:**

Word =believe, synset=think,believe,consider,conceive

similarity(“believe”, synset “[think](http://wordnetweb.princeton.edu/perl/webwn?o2=1&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=think), **believe**, [consider](http://wordnetweb.princeton.edu/perl/webwn?o2=1&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=consider), [conceive](http://wordnetweb.princeton.edu/perl/webwn?o2=1&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=conceive)”) = 72/(118+72+37+10+2), say p1.

P1 is the probability of the word in synset s, synset s / sum of frequencies of synsets of word w.

1. **Step 3 in algorithm:**

Two words w1 and w2 belonging to same synset s1.

Example: similarity (“believe” in synset “ [think](http://wordnetweb.princeton.edu/perl/webwn?o2=1&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=think), **believe**, [consider](http://wordnetweb.princeton.edu/perl/webwn?o2=1&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=consider), [conceive](http://wordnetweb.princeton.edu/perl/webwn?o2=1&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=conceive)”, think in synset “[think](http://wordnetweb.princeton.edu/perl/webwn?o2=1&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=think), **believe**, [consider](http://wordnetweb.princeton.edu/perl/webwn?o2=1&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=consider), [conceive](http://wordnetweb.princeton.edu/perl/webwn?o2=1&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=conceive)”) = p1 \* p2.

P1=probability of believe , synset think, believe, consider, conceive/ sum of frequencies of synsets of word w1

=72/(118+72+37+10+2)=72/240.

P2=Probability of think , synset think, believe, consider, conceive/ sum of frequencies of synsets of word w2

= 277/(277+190+111+8+4+3+2+6)=277/601

So the similarity=p1\*p2

=72/240 \* 277/601=0.3\*0.4608985=0.13826955.

1. **Example for step4:**

W1=believe, w2= believe

Believe is contained in all 5 synsets .It’s similarity is given by

Believe, Believe in synset1 s1=119/240

Believe, Believe in synset2 s2=72/240

Believe, Believe in synset3 s3=37/240

Believe, Believe in synset4 s4= 10/240

Believe, Believe in synset5 s5=2/240

Total similarity=s1+s2=s3+s4+s5

=119/240+72/240+37/240+10/240+2/240

=1

1. **step 5 in algorithm:**

W1=believe W2=repute, LCA1=1/(4+1).

LCA1 is path between synset1 of w1 and synset2 of w2.

P1= w1 in synset1,synset1/ sum of frequencies of synsets of word w1=119/240

P2= w2 in synset1,synset1/ sum of frequencies of synsets of word w2=1

S1=p1\*LCA1\*p2=119/240\*LCA1\*1=0.0991666

P1= w1 in synset2,synset2/ sum of frequencies of synsets of word w1

P2= w2 in synset1,synset1/ sum of frequencies of synsets of word w2

LCA2=1/(3+1).

LCA2 is path between synset1 of w1 and synset2 of w2.

S2=p1\*LCA2\*p2=72/240\*LCA2\*1=0.075

P1= w1 in synset3,synset3/ sum of frequencies of synsets of word w1

P2= w2 in synset1,synset1/ sum of frequencies of synsets of word w2

LCA3=1/(4+1).

LCA3 is path between synset2 of w1 and synset1 of w2.

S3=p1\*LCA3\*p2=37/240\*LCA3\*1=0.03083333

P1= w1 in synset4,synset4/ sum of frequencies of synsets of word w1

P2= w2 in synset1,synset1/ sum of frequencies of synsets of word w2

LCA4=1/(5+1)

LCA4 is path between synset4 of w1 and synset1 of w2

S4=p1\*LCA4\*p2=10/240\*LCA4\*1=0.00694444

P1= w1 in synset5,synset5/ sum of frequencies of synsets of word w1

P2= w2 in synset1,synset1/ sum of frequencies of synsets of word w2

LCA5=1/(7+1)

LCA5 is path between synset5 of w1 and synset1 of w2.

S5=p1\*LCA5\*p2=2/240\*LCA5\*1=0.001041667

Total similarity=s1+s2+s3+s4+s5.

=0.21298604.

1. **Step 6 in algorithm**

**Similarity between word and hypernym**

W1=believe w2=accept

P1=believe in synset1/sum of frequencies of synsets of word w1=119/240

P2=accept in synset1/ sum of frequencies of synsets of word w2=34/117

LCA1=1/(1+1)=1/2

LCA1 is path between synset1 of w1 and synset1 of w2.

s1=p1\*LCA1\*p2=119/240 \* ½ \* 34/117=0.07204416

LCA2 is path between synset4 of w1 and synset1 of w2.

P1=believe in synset4/ sum of frequencies of synsets of word w1 =10/240

P2=accept in synset1/ sum of frequencies of synsets of word w2=34/117

LCA2=1/(2+1)=1/3

S2= p1\*LCA2\*p2=10/240\* 1/3 \* 34/117=0.004036087

LCA3 is path between synset5 of w1 and synset1 of w2.

P1=believe in synset5/ sum of frequencies of synsets of word w1=2/240

P2= accept in synset1/ sum of frequencies of synsets of word w2=34/117

LCA3=1(4+1)=1/5

S3=p1\*LCA3\*p2=2/240\*1/5\*34/117=0.0004843306

Similarity (believe,accept)=s1+s2+s3=0.07656467

**Similarity between word and it’s troponym**

W1=believe, W2=understand

LCA is path between synset1 of w1 and synset4 of w2.

P1=believe in synset1/ sum of frequencies of synsets of word w1=119/(119+72+37+10+2)=119/240

P2=understand in synset4/ sum of frequencies of synsets of word w2=4/(111+29+9+4+1)=4/154

LCA=1/(3+1)

Similarity (believe, understand)=p1\*LCA\*p2

Similarity(believe, understand)=119/240 \*1/4 \* 4/154

=0.4958333\*0.25\*0.0259740

=0.003219697

**Smilarity between derivationally related word and synset s**

W=believer synset s=believer,truster

Similarity(believer, synset “ believer,truster”)=p1=1/2=0.5

**Smilarity between word’s derivationally related word and other word in same synset**

W1=believer w2=truster

Smilarity(believer, truster)=p1\*p2

P1=believer in synset1 / sum of frequencies of synsets of word w1=1/2

P2=truster in synset1/ sum of frequencies of synsets of word w2=1/1=1

Similarity=(believe, truster)=1\*0.5=0.5

**Similarity between derivationally related word and hypernym**

Similarity(believer, abiogenist)=p1\*LCA\*p2

LCA is path between synset1 of w1 and synset1 of w2.

P1=believer in synset1/ sum of frequencies of synsets of word w1=1/2

LCA=1/(1+1)=1/2

P2=abiogenist in synset1/ sum of frequencies of synsets of word w2=1/1=1

Similarity(believer, abiogenist)=p1\*LCA\*p2=1/2\* ½ \* 1=0.25

**Similarity when antonym is given as input to wordnet**

**\*1 Antonym, hypernym of it**

W=disbelieve , hypernym=reject

LCA is path between synset1 of w1 and synset1 of w2.

P1= disbelieve in synset1/ sum of frequencies of synsets of word w1 = 1/1=1

P2=reject in synset1/ sum of frequencies of synsets of word w2=19/40

LCA=1/(1+1)=1/2

Smilarity (disbelieve, reject)=p1\*LCA\*p2=1 \* ½ \* 19/40=0.2375

**Smilarity between Antonym and it’s synset**

W= disbelieve, synset s=disbelieve,discredit

Similarity(disbelieve, synset “disbelieve, discredit”)= 1/1=1

**Antonym and other word**

W1=disbelieve w2=discredit

Similarity (disbelieve, discredit)=p1\*p2

P1=disbelieve in synset1/ sum of frequencies of synsets of word w1=1/1=1

P2= discredit in synset3/ sum of frequencies of synsets of word w2=1/3

Similarity(disbelieve, discredit)=p1\*p2=1\*1/3=0.333333

**Antonym’s derivationally related form and other derivationally related word**

W1=disbeliever w2=discredit

Similarity(disbeliever, discredit)=p1\*LCA\*p2

LCA is path between synset1 of w1 and synset1 of w2.

P1 disbeliever in synset1/ sum of frequencies of synsets of word w1=1/1=1

P2= discredit in synset3/ sum of frequencies of synsets of word w2=1/3=0.333

LCA=1/(1+1)

Smilarity (disbelieve, discredit)= p1\*LCA\*p2=1\* ½ \*1/3=0.1666666

**Antonym and troponym**

Similarity(disbelieve,doubt)=p1\*LCA\*p2

LCA is path between synset1 of w1 and synset1 of w2.

P1=disbelieve in synset1/ sum of frequencies of synsets of word w1=1/1==1

P2=doubt in synset1/ sum of frequencies of synsets of word w2=17/18=0.9444444

LCA=1/(3+1)

Similarity(disbelieve, doubt)=p1\*LCA\*p2=1\* 1/4 \* 17/18=0.2361111

1. **Implementation and Results**

The code is written in java . We have used the Wordnet API (JAWS). The API is downloaded from Princeton site . Using API we query the word net database and get the results . The jaws functions which we have used are as follows:

getSynsets()- this function gets the synsets from wordnet given the seed word.

getWordForms()- this function gets the wordform of the synsets.

getDefinition()-this function gets the definition of the synsets.

gethypernyms()- this function gets the hypernyms of the synsets from the database .

getTroponyms()-this function gets the troponyms of the word.

We are storing the results into the database. The highest similarity obtained between two different words is 0.5. The results are stored in the descending order of similarity. Finally we are displaying the results from the database. We have kept the threshold of 0.01. If the similarity value is less than 0.01 then the word is not in support or dispute dictionary of the seed word. This is because the word is not related to the seed word when the similarity value becomes less than 0.01. This value is determined by us looking at the results. We are successfully able to display 50-60 words in support dictionary and 30-40 words in the dispute dictionary.

**Tools Used**

**Eclipse**

**Wordnet API**

**My SQL**