Progress Report

In the past 6 months, we have been working on matching a quality term (Q-term) to a PATO term. We have tried two approaches, and I will elaborate on them in the following sections.

# First Approach

In the following sections, unless specified otherwise, the similarity of two terms are calculated using perl WordNet::Similarity, and the PATO structure is accessed by Java OWLAccessor.

## Description

The first approach can be described as below:

1. Identify the category term *C* that a Q-term *Q* belongs to. We call *C* **parent** of *Q*, and all other terms under *C* **siblings** of *Q*.
2. Find the PATO match *A* of term *C* using manually matched results.
3. Find all children terms of *A*. We call these children terms **cousins** of *Q*.
4. Find 5 terms most similar to *Q* under the category *C*. We call the 5 terms **closest siblings** of *Q*.
5. Compare each cousin (along with its PATO synonyms OR definition keywords, using maximum) of Q to closest siblings of Q, and select 5 cousins with highest
   1. Maximum similarity with the closest siblings.
   2. Weighted average similarity with the closest siblings.

The 5 cousins are recommended matches of Q.

## Limitation

The limitations of this approach are:

1. The PATO match A of category C could be a leaf node in PATO, which means it does not have any children.
2. Even though the PATO match A is an internal node, the best match of Q could be a term that is not A’s child.
3. If all siblings of Q are not similar to Q, accuracy of the result could be adversely affected.
4. We thought the category and siblings of a Q-term can provide information of the context of the Q-term. However, it turned out that the information of context is not adequate to distinguish two senses of a same term using this approach.

## Solution

To overcome limitation 1 and 2, we have expanded the cousin space to whole PATO terms, which means we have removed step 2 and 3. The logic is embedded in file [approach1runner.pl](wordSenseCode/approach1runner.pl).

To overcome limitation 3, we decided to extract synonyms from dictionary.com as a term’s siblings. The automatic extraction of synonyms can be done by HTML::TreeBuilder and HTML::Scanning. Please see [extractSynonyms.pl](wordSenseCode/extractSynonyms.pl)::getSection.

## Results

[Mapping Result111205.xlsx](results/Mapping%20Result111205.xlsx): Result of the original approach with limitations. The best matches generated by the program sometimes include antonyms. We thought of using other similarity module such as lesk to solve the problem.

[CompareResult\_LeskAndVector111207.xlsx](results/CompareResult_LeskAndVector111207.xlsx): Results for comparison of vector and lesk method. We have decided to use vector since basically lesk does not solve the problems of antonyms.

[folder vector\_wholesapce\_output](results/vector_wholesapce_output): Results after expanding cousin space, but before introducing synonyms from dictionary.com.

[twisted\_allsyn\_wa\_op](results/twisted_allsyn_wa_op): Matches of word “twisted” after expanding cousin space and introducing synonyms from dictionary.com.

## Perl Packages Needed

WordNet::Similarity

WordNet::QueryData

WordNet::InfoContent

HTML::TreeBuilder

HTML::Scanning

# Second Approach

The second approach is inspired by the methodology proposed in this paper (Figure 2):

<http://www.tc.umn.edu/~liux0395/fp143-liu.pdf>

This approach instead uses vector space (file wordvectors.dat) generated by our own corpus. This is enabled by huge-count.pl from TextNSP package and vector-input.pl from UMLS::Similarity package.

## Description

1. Generate vector space using huge-count.pl and vector-input.pl (this could be done by the shell script [genVectors.sh](wordSenseCode/genVectors.sh), if the two perl modules are installed and the two scripts are put in the folder where the shell script resides), and replace the wordvectors.dat used by WordNet::Similarity::Vector module with the new vector space.
2. Use context sentences of the Q-term Q to build vector VQ.
3. Use definition keywords of a PATO term M and its parent to build vector VM in the vector space.
4. Calculate the similarity based on the two vectors VM and VQ.
5. Repeat step 3 and 4 for each PATO term.
6. Select the PATO term with the highest similarity.

The logic is embedded in [approach2runner.pl](wordSenseCode/approach2runner.pl).

## Results

The results [output120306](results/output120306) are not satisfactory. The reason is still under research.

## Perl Packages Needed

WordNet::Similarity

WordNet::QueryData

WordNet::InfoContent

UMLS::Similarity

Text::NSP

HTML::TreeBuilder

HTML::Scanning