Clustering and Classification Augmented with Semantic Similarity for Text Mining

S.Revathi¹, Dr.T.Nalini²

¹ PG Scholar, Department of CSE, Bharath University, Chennai, Tamil Nadu, India.
² Professor, Department of CSE, Bharath University, Chennai, Tamil Nadu, India.

Abstract
Semantic similarity is a way of analyzing the perfect synonym that exists between word-pairs. This measure is necessary to detect the degree of relationship that persists within word-pairs. To compute the semantic similarity that lies between a word-pair, clustering and classification augmented with semantic similarity (CCASS) was developed. CCASS is a novel method that uses page counts and text snippets returned by search engine. Several similarity measures are defined using the page counts of word-pairs. Lexical pattern clustering is applied on text snippets, obtained from search engine. These are fed to the support vector machine (SVM) which computes the semantic similarity that exists between word-pairs. Based on this value obtained from the support vector machine, Simple KMeans clustering algorithm is used to form clusters. Upcoming word-pairs can be classified, after computation of its semantic similarity measure. If it does match with the existing clusters, a new cluster may be created.

Keywords: Semantic Similarity, Similarity measure, Clustering, Classification, Text mining.

1. Introduction
Text mining refers to the knowledge extraction from textual databases or documents. This text mining [18] is different from mining the other types of databases because of its unstructured form and large number of dimensions. Each word in the document is a dimension.

To obtain useful information from the internet, we place the keywords or entities on the search engine, which in turn returns the corresponding results matching the searched keyword or entity. As time crosses, the technique of searching changes. When a word is being searched, much irrelevant information related to the word is also displayed. Semantic Similarity is a fast emerging technique where words are analyzed based on their meaning.

For example, when a user searches for a bank located near their home, may get unwanted results related to the river banks. Maintaining these new meanings together in a dictionary is difficult. It differs based on the usage of that word in that particular phrase.

A new empirical method is proposed to evaluate the semantic similarity that exists between words. The proposed method evaluates the semantic similarity among words based on page counts and lexical pattern clustering based on snippets.

In section II, the related work for CCASS is discussed. In section III, the detailed explanation of CCASS is given. In section IV, the experiment and results of CCASS are discussed. In section V, the conclusion of CCASS is analyzed, followed by the references used.

2. Related Work
Danushka Bollegala [2] has proposed similarity measures using page count returned by the search engine for the given word pair. These similarity measures are modified four popular co-occurrence measures; Jaccard, Overlap, Dice, and PMI (point-wise mutual information). Page-count-based metrics use association ratios between words that are computed using their co-occurrence frequency in documents. The basic assumption of this approach is that high co-occurrence frequencies indicate high association ratios and high association ratios indicate a semantic relation between words.

A technique to calculate similarity between two words is to find the length of the shortest path connecting the two words in the taxonomy [3]. If a word is polysemous then many paths might exist within the two words. Only the shortest path between any two senses of the words is considered for calculating similarity. A problem that it relies is all links in the taxonomy represent a uniform distance.

Resnik [4] proposed a similarity measure using data content. The similarity between two concepts A1 and A2 in the taxonomy as the maximum of the information content of all concepts A that subsume both A1 and A2. The similarity between two words is defined as the maximum
of the similarity between any concepts that the words belong to.

Some work has been carried out on measuring semantic similarity using Web content. Matsuo et al., [5] proposed the application of Web hits for obtaining communities on the Web. They measured the relation between two personal names using the overlap (Simpson) coefficient. This can be decided based on the number of Web hits for each individual name and their conjunction (i.e., AND query of the two names).

Sahami et al., [6] measured semantic similarity between two queries using snippets returned for those queries by a search engine. For each query, they collect snippets from a search engine and represent each snippet as a TF-IDF weighted term vector. Each vector is $L_2$ normalized and the centroid of the set of vectors is computed. Semantic similarity between two queries is then defined as the inner product between the corresponding centroid vectors. They did not compare their similarity measure with taxonomy-based similarity measures.

Chen et al., [7] proposed a double-checking model using text snippets returned by a Web search engine to compute semantic similarity between words. For two words $P$ and $Q$, they collect snippets for each word from a Web search engine. Then they count the occurrences of word $P$ in the snippets for word $Q$ and the occurrences of word $Q$ in the snippets for word $P$. These values are combined nonlinearly to compute the similarity between $P$ and $Q$. This method depends heavily on the search engine's ranking algorithm. Although two words $P$ and $Q$ might be very similar, there is no reason to believe that one can find $Q$ in the snippets for $P$, or vice versa.

3. Clustering and Classification Augmented with Semantic Similarity (CCASS)

Semantic similarity [16] is calculated based on page count and text snippets. Four similarity measures are defined for any 2 words say $P$ and $Q$ based on their page counts namely WebJaccard, WebOverlap, WebDice and WebPMI. The lexical patterns are extracted from the text snippets obtained from the search engine. The patterns are ranked based on their ability to analyze the Semantic similarity. The Support vector machine (SVM) is used to express the semantic similarity value.

After detecting the semantic similarity value, Simple KMeans Clustering algorithm is used to cluster the semantically related words. Thereafter when a new word enters, its semantic similarity value is calculated and it can be easily classified into any one of the existing clusters. If it does not belong to any of the existing clusters, then a new cluster can be created by placing this one member into the new cluster.

![Figure 1: CCASS (Clustering and Classification Augmented with Semantic Similarity)](image_url)

3.1 Similarity measure based on Page Count

Semantic similarity [17] between two words can be calculated based on their page counts. To find the similarity between 2 words say $P$ and $Q$, the page count measure of $P$ AND $Q$ is not sufficient. For example, the page count for “doctor” AND “car” in GOOGLE is 643,000,000. Whereas the page count for “doctor” AND “profession” is 51,200,000. The page count of $P$ AND $Q$ is not sufficient and so the individual page count has also become necessary.

The similarity measures WebJaccard, WebOverlap, WebDice and WebPMI based on the page counts can be defines as shown below:

\[
\text{WebJaccard}(P, Q) = \begin{cases} 
0 & \text{if } H(P \cap Q) \leq c \\
\frac{H(P \cap Q)}{H(P) + H(Q) - H(P \cap Q)} & \text{otherwise.} 
\end{cases} 
\]

\[
\text{WebOverlap}(P, Q) = \begin{cases} 
0 & \text{if } H(P \cap Q) \leq c \\
\frac{H(P \cap Q)}{\min(H(P), H(Q))} & \text{otherwise.} 
\end{cases} 
\]

\[
\text{WebDice}(P, Q) = \begin{cases} 
0 & \text{if } H(P \cap Q) \leq c \\
\frac{2H(P \cap Q)}{H(P) + H(Q)} & \text{otherwise.} 
\end{cases} 
\]
Here $H(P)$ is the page count of word $P$. In our example, the value of $C = 5$. And $N$ is the total number of documents indexed by GOOGLE, for example $10^{10}$.

3.2 Extracting Lexical Patterns from Snippets

The search engine returns text snippets on the result of searching a word. Lexical patterns are extracted from the text snippets. For example, the following is the snippet returned by GOOGLE for the keywords “doctor” and “profession”.

Doctor is a noble profession who serves our country to save human life.

A Lexical pattern extraction algorithm is proposed to extract the patterns from the text snippets obtained from the search engine. Given a set $W$ of synonymous word-pairs, GetSnippets function returns a list of text snippets for the query “$P$” AND “$Q$”. For each “a” returned by GetSnippets function, GetNgrams extract word n-grams, $n=2$, 3, 4, 5. n-grams with perfectly one X and one Y are selected. Let the patterns be selected based on $X^2$ values be GoodPats. Normalizing refers to the process of dividing the count of each pattern by the total number of counts of observed patterns. The similarity measures that are calculated earlier based on the page count of the word-pairs are also uploaded. The final feature vector $F$ is created.

The feature vectors are calculated for each word-pair. The SVM is trained to calculate the semantic similarity for each word-pair. Consider another word-pair ($P'$, $Q'$) and calculate its feature vector $F'$. The semantic similarity $SemSim(P', Q')$ is defined as,

$$SemSim(P', Q') = \text{Prob}(F'|\text{synonymous})$$

which is a posterior probability, that says the feature vector $F'$ belongs to synonymous-word class.

3.4 Simple KMeans Clustering

The semantic similarity values obtained from the support vector machine are clustered based on their probability values. Here, Simple KMeans clustering [20] is used to form clusters. The numbers of clusters are independent of the number of word-pair.

The simple KMeans is a type of partitional clustering. KMeans is an iterative clustering [21] algorithm in which items are moved among sets of clusters until the desired set is reached.

3.5 Classification

After formation of clusters, when a new word enters, it can be easily classified based on the existing clusters. If it does not match with the existing clusters, a new cluster is created and this new member is placed into the cluster.

$$F \leftarrow [PF, \text{WebJaccard, WebOverlap, WebDice, WebPMI}]$$

return ($F$)

A feature vector is created for each word-pair. The snippets for words “$P$” AND “$Q$” are extracted. Words “$P$” and “$Q$” are replaced with X and Y. GetNgrams extracts n-grams, $n=2$, 3, 4, 5. n-grams with perfectly one X and one Y are selected. Let the patterns be selected based on $X^2$ values be GoodPats. Normalizing refers to the process of dividing the count of each pattern by the total number of counts of observed patterns. The similarity measures that are calculated earlier based on the page count of the word-pairs are also uploaded. The final feature vector $F$ is created.

The feature vectors are calculated for each word-pair. The SVM is trained to calculate the semantic similarity for each word-pair. Consider another word-pair ($P'$, $Q'$) and calculate its feature vector $F'$. The semantic similarity $SemSim(P', Q')$ is defined as,

$$SemSim(P', Q') = \text{Prob}(F'|\text{synonymous})$$

which is a posterior probability, that says the feature vector $F'$ belongs to synonymous-word class.

Algorithm 1 : EXTRACTPATTERN

Given a set $W$ of word-pairs, extract patterns.

For each word-pair ($P$, $Q$) $\epsilon$ $W$

do $A \leftarrow$ GetSnippets (“$P$” $Q$”)

$N \leftarrow$ null

For each snippet $a$ $\epsilon$ $A$

do $N \leftarrow N +$ GetNgrams ( $a$, $P$, $Q$ )

Pats $\leftarrow$ CountFreq ( $N$ )

return ( Pats )

Algorithm 2 : GETFEATUREVECTOR($P$, $Q$)

Given a word-pair $P$, $Q$ get its feature vector $F$.

$A \leftarrow$ GetSnippets (“$P$” $Q$”)

$N \leftarrow$ null

for each snippet $a$ $\epsilon$ $A$

do $N \leftarrow N +$ GetNgrams ( $a$, $P$, $Q$ )

SelPats $\leftarrow$ SelectPatterns ( $N$, GoodPats )

$PF \leftarrow$ Normalize (SelPats)

Algorithm 3 : For Classification

Input: set of files $D$, cluster id $C$, clusters

Output: allotting cluster id

Procedure:

Begin

Let $C$ be array of cluster ids

Let $M$ be the array size equals to clusters

for each file $fd$ in $D$

for each cluster

$M[i]= get\_similarity(fd,Ci)$
end for  
cid=\text{find\_max}(M)  
output cid as cluster id for file fd  
update clusters // assign fd to cluster  
with cid as cluter id if cid is new  
form new cluster  
end for  
end

\text{find\_max}(M)
{
\text{Let cid}=0;  
\text{for } i \text{ 1 to size}(M)  
\text{if } m[i]>m[cid]  
cid=i;  
\text{end for}  
\text{if}(\text{cid}=0)  
cid=\text{size}(M)+1; // new cluster id is formed  
\text{return cid};  
}

4. Experiment and Results

4.1 Experimental setup

CCASS is experimented against the Miller-Charles dataset, with 28 word-pair. Their similarity is rated from 0 (no similarity) to 4 (perfect synonym). Miller-Charles data set is a subset of Rubenstein Goodenough's original data set of 65 word pairs. This dataset contains words as pair which are quite similar in meaning. They are formed with a human rating is associated with this data set. Works are also performed with the Rubenstein Goodenough's original data set.

Weka is a collection of open source ML algorithms used for pre-processing, classifiers, clustering, and association rule. Weka is created by researchers at the University of Waikato in New Zealand. It is a Java based tool used in the field of data mining. It uses flat text files to describe the data. It can work with a wide variety of data files including its own ":\text{arff}" format and C4.5 file formats.

**Sample Word-pair**

<table>
<thead>
<tr>
<th>Word-pair</th>
<th>Web Jaccard</th>
<th>Web Overlap</th>
<th>Web Dice</th>
<th>Web PMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cord - smile</td>
<td>0.102</td>
<td>0.108</td>
<td>0.036</td>
<td>0.207</td>
</tr>
<tr>
<td>Rooster-voyage</td>
<td>0.011</td>
<td>0.012</td>
<td>0.021</td>
<td>0.228</td>
</tr>
<tr>
<td>Noon - string</td>
<td>0.126</td>
<td>0.133</td>
<td>0.060</td>
<td>0.101</td>
</tr>
<tr>
<td>Glass - magician</td>
<td>0.117</td>
<td>0.124</td>
<td>0.408</td>
<td>0.598</td>
</tr>
<tr>
<td>Monk - slave</td>
<td>0.181</td>
<td>0.191</td>
<td>0.067</td>
<td>0.610</td>
</tr>
<tr>
<td>Coast - forest</td>
<td>0.862</td>
<td>0.870</td>
<td>0.310</td>
<td>0.417</td>
</tr>
<tr>
<td>Monk - oracle</td>
<td>0.016</td>
<td>0.017</td>
<td>0.023</td>
<td>0</td>
</tr>
<tr>
<td>Lad - wizard</td>
<td>0.072</td>
<td>0.077</td>
<td>0.070</td>
<td>0.426</td>
</tr>
<tr>
<td>Forest - graveyard</td>
<td>0.068</td>
<td>0.072</td>
<td>0.246</td>
<td>0.494</td>
</tr>
<tr>
<td>Food - rooster</td>
<td>0.012</td>
<td>0.013</td>
<td>0.425</td>
<td>0.207</td>
</tr>
<tr>
<td>Coast - hill</td>
<td>0.963</td>
<td>0.965</td>
<td>0.279</td>
<td>0.350</td>
</tr>
<tr>
<td>Car - journey</td>
<td>0.444</td>
<td>0.460</td>
<td>0.378</td>
<td>0.204</td>
</tr>
<tr>
<td>Crane - implement</td>
<td>0.071</td>
<td>0.076</td>
<td>0.119</td>
<td>0.193</td>
</tr>
<tr>
<td>Brother - lad</td>
<td>0.189</td>
<td>0.199</td>
<td>0.369</td>
<td>0.644</td>
</tr>
<tr>
<td>Bird - crane</td>
<td>0.235</td>
<td>0.247</td>
<td>0.226</td>
<td>0.515</td>
</tr>
<tr>
<td>Bird - cock</td>
<td>0.153</td>
<td>0.162</td>
<td>0.162</td>
<td>0.428</td>
</tr>
<tr>
<td>Food - fruit</td>
<td>0.753</td>
<td>0.765</td>
<td>0</td>
<td>0.448</td>
</tr>
<tr>
<td>Brother-mono</td>
<td>0.261</td>
<td>0.274</td>
<td>0.340</td>
<td>0.622</td>
</tr>
<tr>
<td>Asylum - madhouse</td>
<td>0.024</td>
<td>0.025</td>
<td>0.102</td>
<td>0.813</td>
</tr>
<tr>
<td>Furnace-stove</td>
<td>0.401</td>
<td>0.417</td>
<td>0.118</td>
<td>1</td>
</tr>
<tr>
<td>Magician - wizard</td>
<td>0.295</td>
<td>0.309</td>
<td>0.383</td>
<td>0.863</td>
</tr>
<tr>
<td>Journey - voyage</td>
<td>0.415</td>
<td>0.431</td>
<td>0.182</td>
<td>0.467</td>
</tr>
<tr>
<td>Coast - shore</td>
<td>0.786</td>
<td>0.796</td>
<td>0.521</td>
<td>0.561</td>
</tr>
<tr>
<td>Implement - tool</td>
<td>1</td>
<td>1</td>
<td>0.517</td>
<td>0.296</td>
</tr>
<tr>
<td>Boy - lad</td>
<td>0.186</td>
<td>0.196</td>
<td>0.601</td>
<td>0.631</td>
</tr>
<tr>
<td>Automobile - car</td>
<td>0.654</td>
<td>0.668</td>
<td>0.834</td>
<td>0.427</td>
</tr>
<tr>
<td>Midday-noon</td>
<td>0.106</td>
<td>0.112</td>
<td>0.135</td>
<td>0.586</td>
</tr>
<tr>
<td>Gem-jewel</td>
<td>0.295</td>
<td>0.309</td>
<td>0.094</td>
<td>0.687</td>
</tr>
</tbody>
</table>

**Correlation**

| 0.259 | 0.267 | 0.382 | 0.548 |
4.3 Semantic Similarity

<table>
<thead>
<tr>
<th>Word pair</th>
<th>Semantic Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cord – smile</td>
<td>0</td>
</tr>
<tr>
<td>Rooster-voyage</td>
<td>0.017</td>
</tr>
<tr>
<td>Noon - string</td>
<td>0.018</td>
</tr>
<tr>
<td>Glass - magician</td>
<td>0.180</td>
</tr>
<tr>
<td>Monk - slave</td>
<td>0.375</td>
</tr>
<tr>
<td>Coast – forest</td>
<td>0.405</td>
</tr>
<tr>
<td>Monk – oracle</td>
<td>0.328</td>
</tr>
<tr>
<td>Lad – wizard</td>
<td>0.220</td>
</tr>
<tr>
<td>Forest – graveyard</td>
<td>0.547</td>
</tr>
<tr>
<td>Food - rooster</td>
<td>0.060</td>
</tr>
<tr>
<td>Coast – hill</td>
<td>0.874</td>
</tr>
<tr>
<td>Car – journey</td>
<td>0.286</td>
</tr>
<tr>
<td>Crane – implement</td>
<td>0.133</td>
</tr>
<tr>
<td>Brother – lad</td>
<td>0.344</td>
</tr>
<tr>
<td>Bird – crane</td>
<td>0.879</td>
</tr>
<tr>
<td>Bird – cock</td>
<td>0.593</td>
</tr>
<tr>
<td>Food – fruit</td>
<td>0.998</td>
</tr>
<tr>
<td>Brother-monk</td>
<td>0.377</td>
</tr>
<tr>
<td>Asylum – madhouse</td>
<td>0.773</td>
</tr>
<tr>
<td>Furnace-stove</td>
<td>0.889</td>
</tr>
<tr>
<td>Magician – wizard</td>
<td>1</td>
</tr>
<tr>
<td>Journey – voyage</td>
<td>0.996</td>
</tr>
<tr>
<td>Coast – shore</td>
<td>0.945</td>
</tr>
<tr>
<td>Implement – tool</td>
<td>0.684</td>
</tr>
<tr>
<td>Boy – lad</td>
<td>0.974</td>
</tr>
<tr>
<td>Automobile – car</td>
<td>0.980</td>
</tr>
<tr>
<td>Midday-noon</td>
<td>0.819</td>
</tr>
<tr>
<td>Gem-jewel</td>
<td>0.686</td>
</tr>
<tr>
<td>Correlation</td>
<td><strong>0.834</strong></td>
</tr>
</tbody>
</table>

4.4 Simple KMeans Clustering
The clustering algorithm takes the above calculated semantic similarity value as input and separates it into 2 different clusters. 13 word pairs are clustered separately and 15 word pairs are clustered separately.

<table>
<thead>
<tr>
<th>Cluster ID</th>
<th>No. of entities</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13</td>
<td>46</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>54</td>
</tr>
</tbody>
</table>

4.5 Classification
When a new word-pair enters CCASS, its semantic similarity value is calculated. Based on that, it will be placed in the existing cluster or else a new cluster will be created.

According to the results in Table IV, the word-pair “company-stock” yields a semantic similarity value of 0.0867 and will be assigned to the cluster 2. But the word-pair “stock-life” produces a semantic similarity of 0.006. Therefore a new cluster id 3 is generated and this word-pair is assigned to the new cluster.

5. Conclusion
CCASS is a novel method that calculates the semantic similarity between a word-pair. The similarity measures are calculated based on the page counts of the word-pair. Text snippets are extracted from the search engine for the word-pair. Their lexical patterns are extracted and clustered together. These are fed to the SVM to obtain the semantic similarity value as a posterior probability.

Based on the semantic similarity value, clusters are formed to group the semantically similar words. When a new word-pair enters CCASS, it can be easily classified into the existing clusters. If it does not match with the existing cluster, a new cluster can be created for that word-pair.

Acknowledgements:
The authors would thank the management of Bharath University for their support and the Department of Computer Science and Engineering for their encouragement towards the research work.

References
social network extraction system", Proc. of 15th International

for measuring the similarity of short text snippets”, Proc. of

using web search with double checking”, Proc. of the

Efficient Concept-Based Mining Model for Enhancing Text
Clustering,” IEEE Transactions on Knowledge and Data

search engine's index.”, Proceedings of 15th International World

SimilarityDistance”, IEEE Transactions On Knowledge And

to Measure Word-Group Similarity” Proceedings of the 22nd
International Conference on Computational Linguistics, pages

Semantic Similarity Computation between Terms Using Web
Documents” IEEE transactions on knowledge and data
engineering, vol. 22, no.11, November 2010.

the web: A multiontology disambiguation method”, in Proc. of
International Conference on Web Engineering, 2006, pp. 241–
248.

language processing” ACM Transactions on Speech and

“Organizing and searching the world wide web of facts - step
one: the one-million fact extraction challenge”, In Proc. of

on Semantic Similarity Using Information Content”, IJCSI
International Journal of Computer Science Issues, Vol. 8, Issue 4,
No 2, July 2011, pp.364-370.

Similarity metric for solving sparse data problem in ontology
based Information Retrieval System”, IJCSI International

[18] Shaidah Jusoh, Hejab M.Alfawareh, “Techniques,
Applications and Challenging Issue in Text Mining”, IJCSI
International Journal of Computer Science Issues, Vol. 9, Issue 6,
No 2, November 2012, pp 431-436.

[19] Sumit Vashishtha, Dr.Yogendra Kumar Jain, “Efficient
retrieval of Text for Biomedical domain using Expectation
maximization algorithm”, IJCSI International Journal of
Computer Science Issues, Vol. 8, Issue 6, No 1, November 2011,
pp 366-370.

concept based mining model to enrich text clustering”, IJCSI
International Journal of Computer Science Issues, Vol. 9, Issue 4,
No 1, July 2012, pp 417-422.

[21] V.M.NavaneethaKumar, Dr.C.Chandrasekar,. “A consistent
web documents based text clustering using concept based
mining model”, IJCSI International Journal of Computer Science