Applying Latent Semantic Analysis to Classifying Relevance of Forum Messages Using Small-Sized Corpora

Rodolfo RAGA, Jr.
Computer Science Department
Jose Rizal University
10 Shaw Boulevard, Mandaluyong City, Philippines
jundy.raga@gmail.com

Raymund SISON
College of Computer Studies
De La Salle University-Manila
2401 Taft Avenue, Manila, Philippines
sisonr@dlsu.edu.ph

ABSTRACT

This paper presents experiments we conducted in applying Latent Semantic Analysis (LSA) as a tool for supporting analysis of Asynchronous Online Discussion (AOD) transcripts more commonly known as forum messages; in particular, we focused on the task of identifying the relevance/non-relevance of individual message contributions. We implemented a system which employs LSA to classify messages using only very small training corpora. We approach the problem of determining message relevance as a binary (relevant/not relevant) text classification problem relative to an external reference text which serves as the basis of the on-going discussion. We report results that compare LSA’s performance to human coders using Cohen’s inter-coder reliability measure (Kappa). Among others, we also experimented on determining the performance difference between LSA and its predecessor the Vector Space Model (VSM) approach and also determined whether preceding messages in the thread can assist LSA in effectively analyzing the relevance of a message. To the best of our knowledge, our use of very small corpora consisting only of several articles culled from the internet to train LSA for such a task is novel. Our work has the theoretical goal of assessing whether LSA, with its co-occurrence inductive approach, will still perform efficiently for this task even at such small setting.

Key Words: Latent Semantic Analysis, Asynchronous Online Discussion, Forum Discussion.

1. INTRODUCTION

Asynchronous Online Discussions (AOD), also known as forums or threaded discussion boards, are a popular form of web-based computer-mediated communication especially in areas like distributed education and customer support [10]. The advantage brought by this medium is that it frees interlocutors from the limitations of time and space, thereby providing logistic flexibility for communication and learning.

People participate in forums by posting messages, as such; one disadvantage of AODs asynchronous nature is that participants are more prone to get off-topic due to the inherent time-lags in the discussion. Another problem is that manual monitoring and assessment of message contributions is time-consuming and often tedious. For this reason, automated analysis for discussion understanding to enable better information assessment and assistance are much sought for [1].

Although some opinions express that it is probably not feasible or pedagogically appropriate to completely automate the grading of online discussion contributions [11], we believe that a computer assisted pre-analysis rating of the discussion that identify at the very least undesirable distractions, such as contributions that are unrelated or irrelevant to the topic focus, would leverage the manual analysis process.

In this paper, we propose Latent Semantic Analysis (LSA) as a method to determining the relevance of each message contribution. LSA is a statistical tool first introduced for matching queries to relevant documents in information retrieval. Its main strength is its capability to compute semantic similarities between documents beyond simple keyword matching. This improvement is accomplished by holding a simple premise that words that share similar context also have similar vector representations (neighboring words). Exploring this assumption, LSA builds a semantic space where words and documents are represented as vectors. Similarity between two words or documents can then be measured, usually but not always, by the cosine of the angle contained between the vectors representing them in the semantic space. For instance, two sentences that use exactly the same terms with the same frequencies will have a cosine of 1, while two sentences that use no terms that are semantically related, will tend to have cosines near 0 or below. LSA is based on Single Value Decomposition (SVD), a mathematical technique that causes the semantic space to be arranged so as to reflect the major associative patterns in the data, and ignores the smaller, less important influences.

LSA is an attractive method because it is relatively straightforward to train and use. More importantly it has been shown to mimic a number of aspects of human competence/performance [12]. However, because LSA is a statistical theory, most applications which employed LSA relied on large corpora to train it to capture and represent important components of meanings. Our work differs in that we aim to assess LSA’s performance using very small corpora consisting only of several articles culled from the internet. In general, we aim to derive a benchmark on the performance of LSA as opposed to human coders in such small setting. We implemented a system which uses LSA to predict the relevance of each message and compared its classification decision with three human coders using inter-reliability measures computed using Cohen’s Kappa. We approach the problem of determining message relevance as a binary (relevant/not-relevant) text classification problem [15] relative to an external reference text which serves as the basis of the on-going discussion. Among others, we also implemented measures that compare the performance of LSA to the original Vector Space Model (VSM). VSM is the forerunner of LSA; it operates on a bag-of-words basis without the inductive aid of SVD. We also determined whether or not the inclusion of
preprocessing (i.e., stemming) be applied to the documents prior to generating the TDM, by which words (also called terms or tokens) appear in the set of documents. For our purpose, we opted to use the Term Frequency – Inverse Document Frequency (tf-idf) weighting scheme. This scheme assigns the overall weight $W_{ij}$ to each term $(i,j)$ in the TDM using the formula:

$$W_{ij} = tf_{ij} \times (\log_2(n_d/df_i) + 1)$$

where:

- $tf_{ij}$ = is the local term frequency
- $n_d$ = is the number of documents in the collection
- $df_i$ = is the number of documents containing term $i$

Applying term-weighing to the TDM produces an intermediary matrix which we will refer to as the weighted TDM (wTDM). After this, Singular Value Decomposition (SVD) is applied to the wTDM. Applying SVD decomposes the wTDM into three distinct matrices, that is, a $(n \times m)$ matrix $X = USV^T$ where $U (n \times m)$ and $V (m \times m)$ are the left and right singular matrices (orthonormal) respectively, and $S (m \times m)$ is the diagonal matrix of singular values. After obtaining these three matrices, SVD yields a simple strategy to obtain an optimal approximation for the wTDM using a process known as dimension reduction. In simple terms, this process involves setting to zero all but the first $k$-rank highest value in the diagonal matrix. After which, the original matrix is reconstructed by multiplying the $U$, $S$, and $V^T$ matrices. The resulting product is a $k$-reduced matrix $X'$ which is approximately equal to $X$ in the least squares sense, and is of the same rank. That is, $X' = X = USV^T$. This $X'$ can now be treated as a semantic space where words as well as documents can be compared.

The final step is determining the similarity between documents in the semantic space. There are several methods that can be used to generate LSA-derived similarity measure [14]. The most popular and the one we’ve chosen to adopt in our experiments is the cosine similarity measure. The formula we used for computing the cosine between two set of vectors is the dot product of the vectors normalized by the product of their norm $\|\cdot\|$, that is:

$$\text{CosSim}(Q,D) = \frac{\sum W_{Qj}/W_{ij}}{\sqrt{\sum W_{Qj}^2} \sqrt{\sum W_{ij}^2}} \quad (2)$$

Where:

- $Q =$ is a document vector representing the query (known as the pseudo-document)
- $D =$ is a document vector of a document relevant to $Q$
- $w =$ are term weights

3. LSA IMPLEMENTATION

A prerequisite to conducting our experiment is the development of a computer program which implements the LSA algorithm. In our case, we used a web-based interface using Java Server Pages (JSP) and utilized the methods provided in the Java Matrix (JAMA) package for Matrix and SVD computations. Literature indicates that there is no straightforward procedure to determine an optimal rank-reduction for implementing LSA [9, 18] and that other researchers have found it difficult to match the results reported by the original LSA researchers [8]. Bearing this, for our purpose, we resorted to conducting our own tests to cull-out rank parameters and to devise simple means of determining the accuracy of our LSA implementation.

3.1 Testing for accuracy

To validate the accuracy of our LSA implementation we opted to see whether it can mimic the results reported in [13] using the same dataset as shown Table 1.

Our assumption for this test is that, if our system can produce a result comparable to the one implemented by the early authors then we can be assured, at the very least, that our computational implementation is accurate. Our best result for this test is shown in Table 3. For this result, we used a rank $k=2$ and applied only a local (term frequency - tf) weighing scheme (as described in the paper). The output shown on Table 2 is the original result reported by the early authors.

---

1 Stemming refers to the process of reducing word tokens to their stems or root variant.
2 Stopping refers to the process of removing uninformative word tokens from the documents
3.2 Finding a suitable test data

Our next task is to find test data to use in our experiment. A suitable test data, for this purpose, would be a thread with an overall topic-focus based on some external reference text. The messages in this thread will be used to provide instances of posts that will be analyzed whereas the external reference text will be used as the basis for determining the relevancy/irrelevancy of each message. To find this thread, we searched an online computer-related technical forum.

On one forum, we found a thread where participants actively debated the pros and cons of using SQLDataSource, a data-bound control being introduced by Microsoft. This thread was initiated by one of the participants by quoting and providing a link to an online article about SQLDataSource, the contents of this article then served as the basis of the entire discussion. We will be referring to this online article henceforth as the Debates text.

We deemed this thread as suitable for our purpose; since the reference article is quoted in the first message, we can assume that all participants in the discussion first read the article prior to posting and based their contributions on the said article. Table 4 provides some statistical details of this thread.

<table>
<thead>
<tr>
<th>Total number of posts</th>
<th>87</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of participants</td>
<td>16</td>
</tr>
<tr>
<td>Discussion duration</td>
<td>5 days</td>
</tr>
<tr>
<td>Highest number of posted message by a participant</td>
<td>15</td>
</tr>
<tr>
<td>Lowest number of posted message by a participant</td>
<td>1</td>
</tr>
</tbody>
</table>

As already mentioned, the external reference text (in this case, the Debates text) was used as the basis for determining the relevance of each message. In LSA terms this means that the vector of words representing each message will be compared to the vector of words representing the Debates text in some semantic space. To generate this semantic space we culled two more articles from the web which is related to the topic covered by the Debates text but with a slightly different theme. We refer to these two articles as the SQLDTutor text and CompleteTutor text. The descriptions as well as total number of words contained in these two articles are presented in table 5 along with the Debates text.

3.3 Determining an optimal rank dimension

To cull-out an optimal rank dimension for the SVD dimension reduction, we first generated a semantic space using the three text documents enumerated above and ran several cosine similarity measure between the first message of the thread and the Debates text using different k-rank values (k = 1, 2, 3, 4). In conducting the comparison, we opted to directly compare the TDM with the word vector of the query string and recomputed the SVD for each instance as this, according to literature, is the most robust way to produce the best k-rank approximation [3].
Our assumption for this test is that the first message posted is always relevant to the topic focus of the thread; as such, we simply adopted the $k$-value which generated the highest cosine similarity score for this message. For this purpose, we only used a local term-frequency (tf) weighting scheme. Results of this test are shown in Figure 1, as it turned out, a rank of $k=1$, generated the highest value which is an exact 1.0; we however considered this high value to be superficial as all comparison made between the query vector and all the other vectors in the semantic space at this rank resulted to the same score. We therefore adopted the rank $k=3$ which generated the second highest value of 0.99.

Figure 1. Plot of cosine similarity values for message#1 for rank $k$={1,2,3,4}

4. EXPERIMENTS

4.1 Determining Message Relevance

As already mentioned, in determining the relevance of each message contribution we relied solely on computing the cosine of the angle contained between the vector representing each message and the vector representing the Debates text. The computed cosine value for each message is then designated as the message’s relevance score.

If needed, this relevance score can be further classified into several coherence intervals, similar to the category specified by [5]. However, our human coders found it difficult to use such precision relevance scoring and we were forced to fall back to a simple binary (relevant/not relevant) categorical coding approach. Therefore, to be able to compare the systems classification decision with those of our human coders, we treated any message with a computed cosine between 0.1 and 1.0 to be relevant (and assigned it a category value of 1) otherwise it was treated as not-relevant (and assigned a category value of 0). There is nothing empirical about this threshold; we simply deem that a returned cosine angle between 0.1 and 1.0 implies that the message bears some relevance to the reference document regardless of the level of similarity and that any similarity measure below 0.1 is already too minute for our human coders to discern.

4.2 Research Questions

The goal of our experiment is to assess the efficiency of LSA as a tool for delineating relevant and not relevant messages in a thread relative to a text which represents its overall topic focus using only a small set of training corpora. In doing so, we aim to address the following questions:

1. Using only the limited corpora culled from the internet, to what extent is LSA’s message relevance classifying performance agreeable to the decisions made by human coders?
2. In the process of classifying each message as relevant/not-relevant to the thread’s over-all topic focus, how would inclusion of the vector of the preceding message in the semantic space affect LSA’s performance?
3. In such a setup where the dimension of the semantic space used is extremely low, will LSA still exhibit a better performance than its predecessor the Vector Space Model (VSM) which relies on simple direct word overlap?

4.3 Methodologies

To answer the above enumerated questions we executed three distinct LSA analyses run on our dataset, the description of each are as follows:

1. The first run involves a simple direct comparison between the vectors representing each individual message to the vector representing the Debates text in the semantic space. We refer to this experiment run as LSA1.
2. Similarly, the second run involves comparing the vector of each individual message to the vector representing the Debates text, but this time we also embedded the vector of the preceding message in the TDM and included it in generating the semantic space (except for the first message which have no preceding message). We refer to this experiment run as LSA2.
3. The third run is also similar to the first run, but this time we do not apply SVD to the generated wTDM, this has the same effect as computing cosine differences between the two word vectors on a purely bag-of-words basis and essentially mimics the VSM approach. In this setup, after applying the $tf-idf$ weighing scheme to the TDM we directly moved on to comparing the vector of each individual message to the vector representing the Debates text by way of the wTDM. We refer to this experiment run as VSM1.

As already indicated, we also asked three human coders to manually tag the messages in our thread dataset. The coders were instructed to read first the content of the opinion explained in the Debates text, after which they were tasked to classify each message as being relevant or irrelevant to the topic expressed in the text.

To compare LSA’s classification decisions with our human coders, we opted to use Cohen’s Kappa [17] to measure inter-rater reliability. Our idea is that by using Cohen’s Kappa, we maybe able to measure how much of LSA’s dichotomous rating performance aligns to the decisions made by the human coders. Cohen’s Kappa computes the proportion of agreement actually observed between raters after adjusting for the proportion of agreement expected by chance. The formula for Kappa is:

$$Kappa = \frac{O - E}{1 - E}$$

$$\text{(3)}$$

where $O$ is the observed frequency of agreement and $E$ is the expected frequency of agreement.
where:

O $\Rightarrow$ The "observed percentage of agreement" (the proportion of ratings where the raters are in agreement).

E $\Rightarrow$ The "expected percentage of agreement" (the proportion of agreements that would be expected "by chance" between the raters if they were scoring randomly)

5. RESULTS

5.1 Human Coders Inter-reliability

To establish a baseline in analyzing the performance of LSA, we initially determined the inter-rater reliability of our human coders to see whether or not they will fall within the acceptable range of .60 - .70. The three human coders were designated as Coder1, Coder2, and Coder3. The results of applying Cohen’s Kappa to the decisions made by these coders are show in Table 6.

As shown, Coder2 and Coder3 displayed the highest agreement in classifying the messages, obtaining a Kappa value of 0.67. Coder1 and Coder3 are in second place with a Kappa value of 0.65 and lastly Coder1 and Coder2 displayed the lowest agreement with a Kappa value of 0.58.

Although Coder1 and Coder2 displayed below average agreement, we still interpret the above results as indicating that all three coders, to some extent displayed implicit agreement on its decisions with regards its ratings of the messages. Figure 2 provides a superimposed visual mapping of the classification decisions made by all three coders. The vertical axis in the graph represents the decisions made by each coder (1 for relevant and 0 for not relevant) and the horizontal axis represents the sequence of messages (1-87) in the thread.

<table>
<thead>
<tr>
<th>Coder1</th>
<th>Coder2</th>
<th>Coder3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coder2</td>
<td>0.58</td>
<td></td>
</tr>
<tr>
<td>Coder3</td>
<td>0.65</td>
<td>0.67</td>
</tr>
</tbody>
</table>

As shown, Coder2 and Coder3 displayed the highest agreement in classifying the messages, obtaining a Kappa value of 0.67. Coder1 and Coder3 are in second place with a Kappa value of 0.65 and lastly Coder1 and Coder2 displayed the lowest agreement with a Kappa value of 0.58.

5.2 LSA1 agreement with human coders

On the other hand, comparing the decisions made by LSA1 to each of our human coders resulted to data shown in table 7.

<table>
<thead>
<tr>
<th>LSA</th>
<th>Coder1</th>
<th>Coder2</th>
<th>Coder3</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSA1</td>
<td>0.21</td>
<td>0.17</td>
<td>0.2</td>
</tr>
</tbody>
</table>

As shown, the Kappa value of LSA1’s decision for each of the human coders is way below the acceptable range. Our initial interpretation of this data is that there is probably too much disagreement between LSA1’s rating and those of our human coders for LSA to be of any practical use under such conditions. However, we recalled that [12] reported that the rate of gain of vocabulary of LSA was comparable to those of primary school children, this finding was made with a dataset consisting of million of words. If this is so, then we deemed that LSA1’s performance must still be relatively good, perhaps if a child was tasked to do the same classification with so little input, the same inter-rater reliability will be generated if compared to adult coders. Figure 3 shows the graph mapping the classification decision output of LSA1.

In this graph we can clearly see that the concentration of relevant/irrelevant messages displayed in Figure 2 is no longer dominant. For the most part, LSA1 classified more messages as relevant as any of the three human coders, garnering a relevance classification percentage of 43.7%. This is what caused the low Kappa results.

5.3 LSA2 Performance

Seeing that LSA1’s agreement with the human coders is too low, we then asked whether LSA’s performance can be improved simply by increasing the size of the semantic space. We deemed that the best way to do this is by including the vector of the preceding message of the message currently being analyzed within the semantic space. Our initial expectation is that this will improve the performance of LSA, since messages in a thread play certain roles and are related to each other by a conversation context [10, 11]. Furthermore, including the previous message increases the number of tokens in the semantic space, thereby potentially providing more co-occurrence from which LSA can induce deeper relationship. However, after running the experiment, this doesn’t seem the case; the Kappa values produced by LSA2 are shown in table 8:
Table 8. Inter-rater reliability of LSA2 with human coders

<table>
<thead>
<tr>
<th></th>
<th>Coder1</th>
<th>Coder2</th>
<th>Coder3</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSA2</td>
<td>0.06</td>
<td>0.06</td>
<td>0.1</td>
</tr>
</tbody>
</table>

As can be seen, the inter-reliability of LSA2 dropped considerably. Visual inspection of the mapping of LSA2’s classification decisions shown in Figure 4 provides the reason for this very low result.

![Figure 4. Classification Decisions of LSA2](image)

In the above figure, we can clearly see that the number of messages classified as relevant in LSA2 has increased. In fact, LSA2’s relevance classification percentage reached a peak 64.4%. This is not a good thing, as LSA2 seems to have gained a tendency to over-rate the relevance of even those messages considered as irrelevant. This seems to imply a negative effect for incorporating preceding messages in the semantic space (or any other document for that matter), as LSA seems to indiscriminately convert additional co-occurrences into relevance value.

The difference between LSA1’s classification decisions with those of LSA2 can be seen in the overlap graph shown in Figure 5. In the lower horizontal axis (zero axis), messages classified as not relevant (blue diamond) by LSA1 but was considered as relevant (pink squares) by LSA2 can be clearly seen.

![Figure 5. Superimposed classification decisions of LSA1 and LSA2](image)

5.4 VSM vs. LSA

Lastly, we wondered whether LSA’s inductive process still pose an advantage over the direct word-overlap method in a setting where the size of the semantic space is extremely low. This led us to our third and last question: In such small setting, will LSA still exhibit a better performance than the Vector Space Model (VSM) which relies primarily on direct word overlap? VSM1 was meant to provide an answer to this question. The Kappa values produced by VSM1 are shown in Table 9.

Table 9. Inter-rater reliability of VSM1 with human coders

<table>
<thead>
<tr>
<th></th>
<th>Coder1</th>
<th>Coder2</th>
<th>Coder3</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSM1</td>
<td>0.12</td>
<td>0.08</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Besides the low Kappa values shown in the table above, another interesting observation we made in executing VSM1 is that the direct word overlap method of VSM classified 98% of the messages in the thread as not relevant (only two messages was classified as relevant with very low relevance value), the distribution graph for VSM1’s classification decisions are shown in figure 6.

![Figure 6. Classification Decisions of VSM1](image)

The above graph seems to indicate that the original Vector Space Model (VSM) which relies on direct word overlap is having difficulty discerning the relevance value of the messages. This performance is in direct contrast to that of LSA’s inductive approach which still enabled it sufficient flexibility to generate relatively efficient results even at such small setting.

6. INTERPRETATIONS AND FUTURE DIRECTIONS

Although the experiments we conducted are somewhat limited and provisional in many ways. Still, the results we gathered provided some hints regarding the capabilities of LSA and have effectively shown that there is still much room for improvement. For one, we were able to derive an ad hoc benchmark on LSA’s performance as compared to humans in determining the relevance of messages of online discussion transcripts. We believe that the value behind this benchmark is supported primarily by our effort to minimize as much as possible the corpora used to train LSA. To the best of our knowledge, no one has ever attempted to explore this lower boundary of LSA’s performance on similar dataset. However, we realize that more testing is still needed to fully establish this benchmark.

We attribute LSA’s low human-agreement performance in this task to its inability to put each message in its proper context. Even with the aid of SVD, LSA is still prone to misinterpreting overlapped words without considering the over-all theme of each message. As our results have shown, simply adding the preceding message to the semantic space does not remedy this problem. In the future, we plan on merging LSA with other statistical tool (such as Bayesian probability) to enable it to factor context-dependency in its classification decision.

Common perception seems to tie LSA’s performance to large corpora where it can better wield its inductive process [6, 4]. However, it is also by processing large corpora that LSA becomes computationally expensive. Our results have shown that LSA’s performance, for our designated task, remained relatively
stationary, that is, it is still above the VSM and below the human performance level even when trained with very small corpora. Given this, the next question we aim to address is whether it is possible to further improve its performance to make it closer or (acceptably) at par with those of human coders with minimal or no increase in the training dataset. So far, the only related work we’ve found is that described in [16] where the authors focused on applying LSA with small-scale corpora for positioning students in learning networks. In the future we also plan to apply our approach on educational forum transcripts and explore more practical applications. We view educational forum transcripts as distinct from public forum transcripts in that it is more structured and goal-oriented. It also provides an added advantage of being able to verify results through the student’s perspectives.

Up until now, it is still under debate how SVD improves similarity measure in LSA. Our experiments have shown that this feature is critical in enabling LSA to work efficiently even at very small setting where its predecessor, the VSM approach, is already faltering. We believe that analyzing the performance of LSA using small sized corpora is vital in understanding the inner workings of this critical process because such settings enable clearer observations of its performance and output, more so, we also believe as [16] does, that LSA is bound to be utilized more recurrently in similar small-scale settings in future applications.

7. ACKNOWLEDGEMENTS
Java Matrix (JaMa) the java package we used in implementing LSA in our system was developed and provided for free by Mathworks and NIST. Special thanks to the members of the JaMa mailing list community who entertained the authors queries regarding the output of the SVD method of this package, most especially to Cleve Moler for providing such a simple and insightful explanation. Also, gratitude is extended to the human coders Jenny Raga, John Felix Paña, and Rey Kempis (Coder1, Coder2, and Coder3 respectively) for contributing a lot of their time in manually classifying the messages in the forum transcript.

9. REFERENCES