SUMMA-A LSA Integrated Development System

Herbert Baier, Wolfgang Lenhard, Joachim Hoffmann, Wolfgang Schneider
Department of psychology, University of Würzburg
baier,hoffmann,schneider@psychologie.uni-wuerzburg.de
wolfgang.lenhard@uni-wuerzburg.de
SUMMA – A LSA Integrated Development System

Herbert Baier * — Wolfgang Lenhard ** — Joachim Hoffmann — Wolfgang Schneider

* Department of Psychology
University of Würzburg
Röntgenring 11
97070 Würzburg
Germany

* baier,hoffmann,schneider@psychologie.uni-wuerzburg.de
** wolfgang.lenhard@uni-wuerzburg.de

ABSTRACT. Latent Semantic Analysis (LSA) is a technique from the field of natural language processing. Its method is completely statistical in nature, which permits to extract the relations between words based on their common occurrences in texts without any specification of rules or dictionaries. Despite the fact that LSA is only a statistical technique and does not yield real verbal intelligence, it exhibits an astonishing degree of expertise on tasks that afford verbal intelligence and semantic knowledge. The SUMMA platform is a robust and flexible technology that supports the users during the complete LSA development and test cycles. Furthermore, it can be used for productive purposes. The core of the system is the server. Several client applications ratify its effectiveness, versatility and confirm the viability of the LSA technology. The server is platform independent, supports most of the language encodings, and allows clients’ remote communication and extensions to match individual requirements. The present paper discusses how SUMMA attaches the main phases in LSA development and test cycles and presents empirical results from applications.
dans des tâches qui requièrent de l’intelligence verbale et des connaissances sémantiques. La plateforme SUMMA est une technologie robuste et flexible qui assiste les utilisateurs durant le développement et les tests de LSA. De plus, la plateforme SUMMA peut être utilisée à des fins applicatives. Le cœur du système est un serveur. Plusieurs applications client confirmant son efficacité, sa polyvalence et confirment la viabilité de la technologie LSA. Le serveur est indépendant de la plateforme, il supporte la plupart des langages d’encodage, et permet des communications clients à distance ainsi que des extensions correspondant à des besoins spécifiques. Le présent article discute de la manière dont SUMMA articule les phases principales de développement et les cycles de test puis présente les résultats empiriques obtenus lors de diverses applications.

KEYWORDS: Latent Semantic Analysis, natural language processing, verbal intelligence, semantic knowledge, semantic space.

MOTS-CLÉS : de la Sémantique Latente, traitement du langage naturel, intelligence verbale, connaissances sémantiques, espace sémantique.

1. Introduction

Latent Semantic Analysis (LSA) is a technique from the field of natural language processing (Deerwester, et al., 1990). Its method is completely statistical in nature, which permits to extract the relations between words based on their common occurrences in texts without any specification of rules or dictionaries.

Despite the fact, that LSA is only a statistical technique and does not yield real verbal intelligence, it exhibits an astonishing degree of expertise on tasks that afford verbal intelligence and semantic knowledge, as for example multiple choice knowledge tests (Landauer, Dumais, 1997; Lenhard, Baier, Hoffmann, Schneider, 2007), automatic essay grading (Landauer, Laham, Rehder, Schreiner, 1997; Lenhard, Baier, Hoffmann, Schneider, 2007), the measurement of textual coherence and prediction of reader’s comprehension (Foltz, Kintsch, Landauer, 1998), the prediction of knowledge gains of readers on the basis of their background knowledge (Wolfe et al., 1998) and last but not least intelligent tutoring systems (e. g. Caccamise, Franzke, Eckhoff, Kintsch, Kintsch, 2007; Wade-Stein, E. Kintsch, 2004).
The SUMMA platform is an integrated LSA development system based on a server/client architecture. The server is the core of the system and was designed to support multitasking and multi-user capabilities as well as portability (platform independent). Moreover it supports most of language encodings and thus can be used on almost every language. The main tasks on the server are user authentication and authorization, corpora administration, generation and weighting of frequency matrices, singular values decomposition (SVD), generation of semantic spaces and calculation of word/text similarities. The Lanczos interactive algorithm (Lanczos, 1951) is used for decomposing the singular values. The server supplies an API (Application Programming Interface) that allows third party software (clients) to communicate locally. The remote communication is supported via remote method invocation using Java RMI that extends the server API. Thus, a client can invoke a remote object in the server in an easy and standardized way.

There are several client applications already implemented, such as the server administration web interface (Lenhard, Baier, Schneider, Hoffmann, 2006), a system for automatic essay scoring of student writings in university lectures and laboratory prototypes of conText, an intelligent tutoring system aimed at fostering text comprehension in students (Lenhard, Baier, Hoffmann, Schneider, Lenhard, 2007) and other future ideas like automatic classification of electronic texts and messages. This paper is organized as follows. In Section 2 the SUMMA platform is described. Next, an alternative weighting function to the traditional log-entropy is presented in the Section 3. Some empirical results are explained in Section 4 and an example of an application to search plagiarism on a semantic basis is shown in Section 5. Finally, in Section 6 comments and foreseen improvements to the SUMMA platform are exposed.

2. The SUMMA Technology

The robustness, flexibility and completeness of the SUMMA platform have improved the research and development on LSA technology. Many research groups have profited and still profit from it, as for example the Department of Psychology of the University of Würzburg in
Germany and the Laboratory of Computer Sciences Paris 6 (LIP6) in France. Next, the server and some clients are described.

2.1. Server Features

The SUMMA server is a pure Java application and hence Platform Independent. It supports multitasking allowing multiple processes to be executed seemingly simultaneously and without interfering with each other. This is the basis to allow concurrent access by multiple users, which can log into the system and perform their tasks. In order to restrict these actions a set of security rules are integrated. These include authentication and authorization. In the authentication step the identity of the user trying to access the system is validated against her / his credentials user name and password. When a user logs on to the server, (s) he is associated with a security identity for the duration of her / his session. This security identity is returned to the user and is required in further communications. Once the user is authenticated and begins to interact with the server. The authorization takes place, which restrict the user to the actions that the (s) he is allowed to execute. The authorization is done on a per-method basis. Server modules specify their own security policy for the provided services. This accomplishes read, write and / or execute rights. Hence, authorization to a service in a module is granted to a user if (s) he owns the respective rights to the required module.

LSA is based on text corpora with each single text usually being split into fragments, as for example paragraphs (Wiener-Hastings, 1999). A text fragment is called document. The server provides a set of tools to support their administration with each document being stored within a single file using UTF-8 encoding. The text corpora on the server are organized in tree structures, with each node representing a corpus that can contain documents or not. Corpus in these trees can be created, edited, moved or deleted. Once a corpus is selected, new documents can be added or existing documents can be edited or removed. A set of documents can be imported from a zip file to a corpus. On the other hand the documents of a corpus can also be exported to a zip file. The
The first step in the LSA process is to building a frequency matrix from the information stored in the corpora. The columns of this matrix contain the documents and the rows contain the different words. Each cell holds the frequency of a specific word in a specific document. Depending on the size of the corpora, the included text material and the language, most of the cells are empty. Compared to English, this is especially true for languages with a high number of inflected words and rich compounding as for example French and German, at least when no lemmatization (reduction of inflected words to their dictionary form) or filtering of infrequent words is applied to the raw text material. A medium sized corpus in German language (e.g. 5 million words in 50 000 documents) usually results in a sparse matrix with a density below 0.05%. Thus, the server does not store the complete frequency matrices but only the non empty cells. Segmentation and mapping methods are supported in the building process of frequency matrices. The first allows to extract the words from the documents and the second to perform different transformations, e.g. to convert the words to lower case, to remove stop words, to lemmatize words and so on. In order to create a frequency matrix, a segmentation method and the desired corpora have to be selected. Furthermore, mapping methods and restriction to a word interval (e.g. the first 200 words) in the documents can also be used. All the building information is saved and can be used in further analyzes. The frequency matrices are organized in a tree structure as well. Different tools are supplied for statistical analysis of the words, documents and corpora in the generated matrices. The matrices can be edited, moved or deleted.

These frequency matrices already comprise all the information that is subsequently necessary for text processing. Nevertheless, they are too huge to reasonably apply them to similarity judgments since they contain irrelevant information (“noise”). To reduce the noise, several steps are necessary. First of all, words with either very low or very high frequency are usually filtered out. To this end the statistical tools to analyze the frequency matrices are helpful. The server provides filters for word, texts and cell values. In a next step a weighting algorithm
is applied to the matrix (see Nakov, Popova, & Mateev, 2001) in order to emphasize words with a specific meaning. Several weighting algorithms are supported as for example log-entropy. This process is called weighting frequency matrices and can be applied on an already weighted frequency matrix as well. A weighted frequency matrix is saved as a child of the respective source matrix in its tree. Again all the building information is saved and can be used in further analyzes with several helping statistical tools.

Following the LSA development process, frequency matrices (weighted or not) are decomposed via Singular Value Decomposition (SVD) similar to the process in a Principal Component Analysis (PCA). Contrary to the eigenvalue decomposition in PCA, where a decomposition of the square matrix of covariances is performed, the SVD used in LSA is the decomposition of a rectangular matrix of weighted term frequencies (for a mathematical description see Berry, Dumais, & O’Brien, 1995; Martin, & Berry, 2007). The decomposition results in three orthogonal partial matrices: A term matrix (comparable to the factor values in PCA), a document matrix (comparable to the factor loads in PCA) and a diagonal matrix with the singular values (comparable to eigenvalues in PCA). By reducing the number of extracted dimensions to a minimum, noise is reduced and the amount of data and memory consumption is downsized. Contrary to PCA, there is no criterion about how many dimensions should be extracted. Numbers of 300 to 1500 dimensions turned out to work best (Dumais, 1991; Graesser et al., 1999; Nakov, 2000; Wild, Stahl, Stemsek, Neumann, 2005). As the SVD is extremely resource costly in case of complete decomposition of the frequency matrices, it is highly recommended to use the Lanczos (1950) iterative algorithm, which is efficient to decompose very large and sparse rectangular matrices when a predefined number of dimensions is demanded. The SUMMA server implements this algorithm.

The results of the SVD establish an n-dimensional orthogonal space (“semantic space”), where the terms and documents are distributed in the same space according to their common usage. Thus, the vector of a term or document partly represents its semantic content. The position within the semantic space reflects the relation of the meaning to other words or documents. As a result, words occur near to each other in the
semantic space if they are often used in the same contexts, no matter whether they are or not actually used in the same documents (higher order cooccurrences; Lemaire, & Denhière, 2004; Kontostathis, & Pottinger, 2002). Spaces to be used in further similarity calculations can be generated by the server with a desired number of dimensions (lower or equal than the dimensions obtained in the SVD phase) and the corresponding information.

The SUMMA server can work with different semantic spaces in parallel performing similarities. To this, diverse measure functions are implemented, e.g. cosine, Euclidian distance, combination of Cosine and Euclidian distance, combination of Cosine and vector length, and so on. New text material can be compared by projecting it into the semantic space, a process called “folding in” that is also supported by the server. Simply speaking, the words of the new text are filtered and weighted in the same way as the original text corpus and multiplied by the respective word vectors in the semantic space. The singular values can be utilized in the weighting process. Subsequently the vectors of the words are summed up to a new vector whose length and direction represent the meaning of the new text. To find similar words or documents within the semantic space, new text material is folded into the semantic space and words and/or texts are retrieved, that meet predefined proximity thresholds. Words and/or texts from a semantic space can also be used to find their nearest neighbor within the same space. The document search can also be restricted to a subset of corpora of the original semantic space corpora. Nevertheless, direct comparison among words, documents and/or new folded in text material is also supported.

In contrast to other methods of automated text analysis, LSA is able to categorize semantically related texts as similar, even when they do not share a single word. For example, the two sentences “A penguin is a bird that lives on fish and krill” and “Penguins are birds, which eat crabs and fishes” have a cosine of .763, despite the fact, that they do only share the word “and” (computation done with word material in German language). On the other hand, the first sentence has only a cosine of .563 with the sentence “A whale is a marine mammal that lives on fish and krill”, although there is a large word overlap between the two
sentences (demonstration available under Lenhard, Baier, Schneider, & Hoffmann, 2006).

Despite its elegance, there are limitations to LSA (see Kintsch’s article in this issue). One of the main drawbacks is the lack of syntax, word order and negation. For an LSA-system “heaven” and “hell” mean more or less the same, because the two concepts are highly interconnected to each other. As a result, LSA-systems in general do not work well on topics and tasks that highly rely on argumentation structure and logic. Moreover LSA usually performs better on texts containing multiple sentences compared to short answers (e.g., only single sentences; Landauer, Foltz, & Laham, 1998).

2.2. Architecture

The SUMMA server is the core of the system and supports following main features: multitasking and multi-user capabilities as well as portability (platform independent). Its structure is modular with the communication between the modules being hierarchical. The modules are organized in levels (see Figure 1):

– Only one module belongs to the first level, the Kernel.
– Every module must communicate with exactly one module in a lower level with the exception of the Kernel.
– No communication is allowed between modules at the same level.
– A module can communicate with any module at a higher level.

This modular hierarchical concept allows extensions of the server in an easy way. Only a module in a lower level can supply an interface for the communication. The Kernel module is the heart of the server being the central point for data sharing and control of remainder modules. It is responsible for the configuration management, logging, authentication and authorization. The API module is the application programming interface, which allows third party software to communicate locally with the server. The RMI (Remote Method Invocation) module is an extension to the API that allows remote communication using Java RMI. Thus, a client can call a remote object in the server in an easy and standardized way. The persistence module is responsible for the data
The inter-modules communication is a rooted tree with the root being the Kernel module.

The modules corpora, matrix and space are an extension of persistence module that manages the text corpora, frequency and weighted matrices, and semantic spaces, respectively. The SVD module performs the Singular Value Decomposition. The similarity module deals with the similarities calculations that are extended by search and comparison modules. The search module finds similar words or documents within the semantic space and the comparison module performs direct association among words, documents and/or new folded in text material.

The server includes a logging system that can be set on different levels: debug, info, warn, error, or fatal. Hence, activities on the server can be traced according to the required necessities.

### 2.3. Communication and Clients

The SUMMA server supports two kinds of communications to third party software: local and remote. Local communications use the API module (Application Programming Interface) and remote communications use the RMI module (Remote Method Invocation). Several client
applications were developed such as:

– The server administration web interface that assists the users during the complete LSA development and test cycle (Lenhard, Baier, Schneider, Hoffmann, 2006)

– ASSIST that is a system for automatic essay scoring of student writings in university lectures

– Laboratory prototypes of conText, an intelligent tutoring system aimed at fostering text comprehension in students (Lenhard, Baier, Hoffmann, Schneider, Lenhard, 2007)

– Document classification into categories, which takes in account the category’s judgments (Ahat, Lenhard, Baier, Hoareau, Jhean-Larose, Denhière, 2007)

Nowadays, some applications are running in the same computer than the server and other applications are running remote in other countries. The confidentiality and integrity protection of the data transfer between the server and clients is not a responsibility of the server but can be performed through cryptographic services like SSH.

2.4. Extensibility for Individual Requirements

The server was designed to allow extensions to match individual requirements that can be attached to it via plug-ins. For example there is a segmentation plug-in that extracts the words from the documents matching a regular expression that considers only alphabetic characters. As a consequence, this plug-in does not work for languages like Chinese, but new plug-ins can be created and registered to the server that receives as parameter a document and returns its array of words using the specific Chinese rules.

Following functions on the server support plug-ins:

– segmentation - allows to extract the words from documents.

– mapping - performs different transformations, for example converts the words to lower case, removes stop words, lemmatizes words.

– weighting - emphasizes words with a specific meaning in the frequency matrices, for example log-entropy.
The server administration Web interface is subdivided into two main interfaces: system and LSA. The system interface concerns pure server administration tasks like server status, user administration, server shutdown and so on. The LSA interface concerns the LSA development platform. Both interfaces require from the beginning that the user herself / himself authenticate and authorizes on the server. Currently, the interface supports internationalization for English, Germany and French. Internationalizations for other languages can be implemented easily.

![conText](image)

Figure 2: The welcome site of the LSA interface. The user is authenticated and authorized.

After a successful login and depending on his / her authorization rights the user can chose between two main function modules: semantic space and corpora. In the semantic space module the user can select semantic spaces and perform similarity judgments. The similarity judgments support all measure functions from the SUMMA server, which
allow a selection of the desired space dimensions. Following similarity judgments are implemented:

- nearest neighbors: given a word retrieves a desired number of semantically similar words. A regular expression filter can be applied to the returned words.
- word comparison: given two words returns their similarity judgment.
- text comparison: given a set of new texts, project them into the semantic space and returns their similarity judgments.
- synonym test: performs systematic tests with lists of synonyms, analogies and so on, as they are used in, for example, intelligence tests.
- nearest documents: given a document in the selected space retrieves a desired number of semantically similar texts.
- neighbors of pseudo documents: given a new text, projects it into the semantic space and retrieves a desired number of semantically similar documents. Several filter options are implemented.
- comparison of documents: given two documents in the semantic space, it returns their similarity judgment.
- comparison of pseudo document and document: given a new text and a document in the semantic space, project the new text into the semantic space and returns their similarity judgment.

The corpora module covers all functionalities needed to create semantic spaces. It is subdivided into administration, matrices and mappings. The administration sub module is concerned with the corpora administration such as create, edit, move, delete, import (upload), export (download) and so on (see Section 2).

The matrices sub module supports the LSA development process from creating a frequency matrix up to generating the semantic space. When creating a frequency matrix, a segmentation method and the desired corpora have to be selected. Furthermore, mapping methods and restriction to a word interval in the documents can also be used. Following the LSA development process, weighting frequency matrices can be created using different weighting algorithms and filters for word, texts and cell values. Finally, the frequency matrices (weighted or not) can
Figure 3: The administration sub module of corpora. On the left side are the possible actions and on the right are the corpora. The numbers mean the number of documents in the respective corpus.
be decomposed via Singular Value Decomposition (SVD) and the semantic spaces with the desired dimensions generated. Several tools are supplied for statistical analysis. The matrices can be edited, moved or deleted.

The last sub module mapping allows performing tests on the registered segmentation and mappings plug-ins.

### 2.6. Global versus Local Weighting Functions

The main goal of the weighting functions is the contrast accentuation, *i.e.* the emphasis of words with a specific meaning. The most common function is log-entropy, which first estimates the entropy of the words and then the logarithmic frequency value of the words is multiplied by the respective entropy. More precisely, for an $n \times m$ matrix representing $n$ words and $m$ documents (with $a_{ij}$ being the $i$th $j$th element of the matrix) the log-entropy function is defined as follow:

- **Entropy:**
  \[
  \delta_i = 1 + \sum_{j=1}^{m} p_{ij} \frac{\ln(p_{ij})}{\ln(m)} \cdot p_{ij} = \frac{a_{ij}}{\sum_{u=1}^{m} a_{i,u}}
  \]

  The entropy $\delta_i$ is used to generate the weighted matrix and to fold in new texts (compound vector of new texts):

  - **Weighted matrix:** $a'_{ij} = \ln(a_{ij} + 1) \cdot \delta_i$
  - **Fold in:** $\overrightarrow{v'} = \sum_{i} (\ln(f_i + 1) \cdot \delta_i) \cdot \overrightarrow{v_i}$

  where $f_i$ is the frequency of the word $i$ in the new text and $\overrightarrow{v_i}$ is its vector in the semantic space.

  We call this function global, since it does not take into consideration the size of the documents (number of words) that the words belong to. An alternative to this function could be to use the entropy but before the weighted matrix must be generated or a new text to divide the frequency of the words by the length of the respective documents must be folded.
Figure 4: The matrices sub module for a selected weighted (normalized) matrices. a) On the left are the possible actions and on the right the selected weighted matrix. b) A word statistic for the selected matrix is shown.
Figure 5: The matrices sub module for a selected SVD-matrix with 300 dimensions. a) On the left are the possible actions and on the right the selected SVD-matrix. b) The singular values of the selected matrix are shown.
Hence, all values are between 0 and 1, where words that often occur will have larger values. Since the values are restricted to the interval 0 and 1, the log function is not more applicable. Furthermore, the values will usually be very small (tend to 0). The $n$th root function is a good option, since it can increase the contrast and still return values between 0 and 1. We call this function local weighting.

- Weighted matrix: $a'_{ij} = \left(\frac{a_{ij}}{|d_j|}\right)^{1/r} \cdot \delta_i$
- Fold in: $\vec{v} = \sum_i \left(\frac{f_i}{|d_j|}\right)^{1/r} \cdot \delta_i \cdot \vec{v}_i$

Empirical results using the log-entropy and 2, 4, 8 and 16 root-entropy functions shows that the $4^{th}$ root-entropy function is more reliable.

3. Empirical Results

The empirical results explained in this section were performed for the German language with exception the last one (Text Classification) that was performed for English. The server was running on a PC with a Pentium IV processor with a clock frequency of 3.2 GHz, 3 GB of RAM.

3.1. Synonym test

249 students grade 5 to 10 and 44 graduate students in biology accomplished a knowledge quiz on animal classification. They should assign the species (amphibian, fish, insect, reptile, mammal, bird, spider) of 90 different species (eel, eagle, alligator,...). The used semantic space is based on texts from the areas of biology, geology and geography from textbooks, encyclopedias and Internet sites. The texts were automatically divided into sections, stop words filtered and the characters converted to lower case. All of the words that occurred less than three times were eliminated. The frequency matrix included 37,773 documents with 83,369 different words (size of the entire corpus 2,178,432 words). The log entropy weighting function was applied and the SVD was running up to 1,000 dimensions. The calculation of the
semantic space took 125 min and the answer of the quiz a fraction of a second.

The LSA-based system accomplishes a hit rate up to 96.7% (310 dimensions) and performed highly significant better compared to the students. It showed an equal performance compared to the graduate students (Lenhard, Baier, Hoffmann, & Schneider, 2007).

3.2. Automatic Essay Scoring

The aim of the study was to verify the correlation between the scoring of open answers given by humans and by a LSA-based system. A sample of 8 exercises from psychology intermediate examinations of 40 students was used. The semantic space was based on the texts of 14 textbooks of psychology. The stop words and words with an occurrence of once were filtered. In the semantic space remained 66,611 different words in 27,688 documents with a total of 1,316,599 words. The calculation of a semantic space for 1,000 dimensions took 80 min. The correlations between human scores and LSA yield high to very high correlations varying from 0.604 to 0.874 (Lenhard, Baier, Hoffmann, & Schneider, 2007).

3.3. Summary scoring

42 Students summarized two expository texts from biology. The length of a summarized text should be between 10% and 20% of the original text. The same semantic space from the synonym test (Section 4.1) was used. A rating scheme was defined and 3 different humans rated the summaries.

The mean correlation between the humans was 0.688 for text 1 and 0.816 for text 2. The mean correlation between the humans and LSA was 0.629 for text 1 and 0.640 for text 2. Therefore, LSA showed to be appropriated for automatic generation of content feedback (Lenhard, Baier, Hoffmann, & Schneider, 2007).
3.4. Text Classification

Texts should be classified into predefined categories. To this end, the 4 Universities Data Set WebKB was used. This data set contains web pages (documents) collected from various computer science departments. The 8,282 pages were manually classified into the categories course (930), department (182), faculty (1,124), project (504), staff (137), student (1,641) and others (3,764). The data set was subdivided into 5 classes with a class containing either the documents from the universities Cornell (867), Texas (827), Washington (1205), Wisconsin (1263) or from the remainder universities (4,120).

<table>
<thead>
<tr>
<th>Class</th>
<th>course</th>
<th>department</th>
<th>faculty</th>
<th>others</th>
<th>project</th>
<th>staff</th>
<th>student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cornell</td>
<td>44</td>
<td>1</td>
<td>34</td>
<td>619</td>
<td>20</td>
<td>21</td>
<td>128</td>
</tr>
<tr>
<td>miscellaneous</td>
<td>686</td>
<td>178</td>
<td>971</td>
<td>693</td>
<td>418</td>
<td>91</td>
<td>1083</td>
</tr>
<tr>
<td>Texas</td>
<td>38</td>
<td>1</td>
<td>46</td>
<td>571</td>
<td>20</td>
<td>3</td>
<td>148</td>
</tr>
<tr>
<td>Washington</td>
<td>77</td>
<td>1</td>
<td>31</td>
<td>939</td>
<td>21</td>
<td>10</td>
<td>120</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>85</td>
<td>1</td>
<td>42</td>
<td>942</td>
<td>25</td>
<td>12</td>
<td>156</td>
</tr>
</tbody>
</table>

Table 1: The 4 Universities Data Set WebKB.

Each class was tested against the remainder classes as follow: The documents of a class were removed from the data set and out of the remaining documents a semantic space was build. The semantic space was then used to classify the previously removed documents.

As long as the category ‘others’ remains in the data set, very bad classification results were obtained. After removing this category and using the similarity function cosine about 70% of the documents were classified correctly (the text was folded and it was classified in the same category as its nearest document in the semantic space using the cosine similarity function). Using the fact that the vector lengths also comprise an influence in the semantic meaning, the following similarity function was implemented:

1) The 10 nearest documents using the cosine function were selected.
2) The length of the selected document were normalized with respect to the maximum length in the selection set: $n_i = l_i / \max \{l_i\}$, where $l_i$ is the length of the $i$th selected document.

3) The text was classified in the same category as the pre-selected document with maximum measure function $\cos_i \cdot n_i$, where $\cos_i$ is measured between the text and the pre-selected document.

Using this new measure function 82% of the texts were right classified.

Figure 6: The text classification results considering the class others, without the class others using the measures functions cosine, and without the class others using the cosine combined with vector length.

3.5. Application: Plagiarism-Detection on a Semantic Basis

The client ASSIST is a lecture assisting system for the easy administration and submission of course task solutions. Two main modules use LSA features: the automatic essay scoring of student writings and the plagiarism detection.

In order to search for plagiarism, all student writings have to be compared with each other. Let $n$ be the number of essays. This means, at least $n \cdot (n - 1)/2$ comparisons are required. The cost of a compari-
son among two texts to find their similarity depends on the underlying algorithm. Let m be maximum size of the texts. A variation of the Smith-Waterman algorithm, which is a well-known algorithm for performing local sequence alignment that uses dynamic programming, can be used to calculate the similarities. However, this algorithm is fairly demanding on time and memory resources. Its running time and space is $O(m^2)$. This yields a total running time $O(m^2) \cdot n \cdot (n - 1)/2$.

Another approach will be to use LSA for the text similarity comparisons. First the student writings are folded in with a running time $O(nm)$. Then their vectors are compared to get a rank list, which yields a running time $O(nm) + n \cdot (n - 1)/2$. This results in a large speed up if we are considering student classes with about 500 students, which mean about 124,750 single text comparisons. Furthermore, LSA does not only consider the common words but also the contextual similarities. In practice it has shown to be a powerful tool to detect plagiarism. After the plagiarism rank list is generated, ASSIST uses on demand the Smith-Waterman or word overlap algorithm to show the texts.

<table>
<thead>
<tr>
<th>User Name 1</th>
<th>User Name 2</th>
</tr>
</thead>
</table>

Figure 7: The ASSIST Plagiarism view using LSA for text similarities and Smith-Waterman algorithm for similarity word level.
4. Discussion and Improvements

The SUMMA server is a powerful tool for LSA users. Since December 2005 it is running stably without significant upgrades. It supports the users during the complete LSA development and test cycles, and can be used for productive purposes as well. Several client applications confirm its profitableness. However, some improvements remain to be done. At the beginning the server was foreseen to be used by a restricted number of users and its configuration should not be changed frequently. Hence, users database and server configurations like running semantic spaces cannot be changed on line. The respective xml configuration file has to be updated and the server restarted. Furthermore, a weakness of the server is it’s memory utilization. The semantic spaces are all loaded in the main memory as well as the SVD computations. Another restriction is that only remote clients using Java RMI are supported. A new version of the server is planned, which will overcome these problems. The new version will use J2EE technology and the semantic spaces will be recovered on demand from a database. The SVD process will also be persisted on a database. The clients will also connect using simple object access protocol SOAP, this mean, clients from other programming languages like .NET, C, C++ and Perl will also be supported.
References


SUMMA-A LSA Integrated Development System


