An LSA Approach in DEFT'07 Contest

Murat AHAT[1], Wolfgang LENHARD[2], Herbert BAIER[2],

Vigile HOAREAU[1], Sandra JHEAN-LAROSE[1] et Guy DENHIÈRE[1]

1Équipe CHArt « Cognition Humaine et Artificielle » - E.A. 4004

EPHE - CNRS
41 Rue Gay Lussac
75005 Paris
France

murat.ahat@google.com

2 Department of Psychology
Universität Würzburg
Röntgenring 10
97070 Würzburg
Germany

wolfgang.lenhard@uni-wuerzburg.de

1 Introduction

In this DEFT'07 contest, we are presented with texts grouped in four categories which include judgments as well. The task is to determine to which judgment the texts belong within a category, while the category of the text is already given. Simply speaking, the task is Document Classification. To achieve this goal, we have built a simple automatic/semi automatic process based on Latent Semantic Analysis (LSA), which produces encouraging results. This process takes into account what the psychology of comprehension has called the mental representation of a document to propose an algorithm for categorization. This approach is based on the achievements accomplished by cognitive psychology in text comprehension and its simulation (Denhière, Lemaire, Bellissens et Jhean-Larose, 2005). Following sections will describe how we have built the automatic process, and the results we have obtained. Finally, the results are discussed.

2 Automated Document Classification

Automated processing is vital in classification of a large number of texts. Without an automated process, classification of texts would be a tedious and boring job. We have based our automated process on LSA - Latent Semantic Analysis. LSA is a statistical technique from the field of natural language processing (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990). It permits to extract the relations between words based on their common occurrences in texts. Its procedure is completely statistical in nature. That means the word meanings reflected by the cooccurrences are extracted completely without any specification of rules or dictionaries.

2.1 Extracting the Latent Semantic Content of Written Language

LSA is based on text corpora with each single text usually being split into fragments, as for example paragraphs (Wiemer-Hastings, 1999). The information stored in the corpora can formally be represented by building a frequency matrix. The columns of this matrix contain the fragment

indices 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 cells are empty. Compared to English language, 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 .05%.

This huge frequency matrix already comprises all the information that is subsequently utilised for text processing. Nevertheless, it is too huge to reasonably apply it to similarity judgements because it contains of 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. In the next step a weighting algorithm is applied to the matrix (Nakov, Popova, & Mateev, 2001) in order to emphasize words with a specific meaning. Finally, the matrix is decomposed via Singular Value Decomposition (SVD) similar to the procession in a Principal Component Analysis (PCA). Contrary to the eigenvalue decomposition in PCA, where a decomposition of the square matrix of covariances is done, the SVD used in LSA is the decomposition of a rectangular matrix of weighted term frequencies (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 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, Stermsek, & Neumann, 2005). As the SVD is extremely resource costly in case the frequency matrices are decomposed completely, it is highly recommended to use the Lanczos (1951) algorithms, to do an abridged computation with a predefined number of dimensions. The SVD is an extremely resource costly process. However, the Lanczos (1951) iterative algorithm can be used to decompose very large and sparse rectangular matrices, when a predefined number of dimensions is demanded.

Eventually, the results of the SVD establish an n-dimensional orthogonal space ("semantic space"), where the terms and documents are distributed according to their common usage. Thus, the vector of a term 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 actually used in the same documents or not (higher order cooccurrences; Lemaire, & Denhière, 2004; Kontostathis, & Pottenger, 2002).

2.2 Using LSA for Similarity Judgements and Information Retrieval

The SVD is a relatively resource intense computation process. Once it is finished, it enables high-performance similarity judgements between words or texts. There are several possible similarity measures like the Euclidian distance or the cosine of the angle between two vectors. The cosine turned out to be a robust similarity measure (Landauer, Laham, Rehder, & Schreiner, 1997; Rehder, Schreiner, Wolfe, Laham, Landauer, & Kintsch, 1998). Moreover, it yields a simple interpretation because it can be used just like a linear correlation.
New text material can be compared by projecting it into the semantic space, a process called "folding in". 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. 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 documents within the semantic space, new material is folded into the semantic space and texts are retrieved, that meet predefined proximity thresholds.

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).

Altough its elegance, there are limitations to LSA. One of the main points 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).

Despite the fact, that LSA is only a statistical technique and does not yield real verbal intelligence, LSA nonetheless 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, in press), automatic essay grading (Landauer, Laham, Rehder, & Schreiner, 1997; Lenhard, Baier, Hoffmann, & Schneider, in press), 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; Lenhard, in submission).

2.3 Technical Infrastructure

The LSA-platform is 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 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 standard way.

There are several client applications 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).

The computation time for a semantic space varies with the density of the frequency matrix and the distribution of the words. The decomposition of the biggest DEFT'07 corpus (Débat) and extraction of 300 dimensions took 18 min 12 sec for instances, whereas the smallest (Articlè) afforded only 39 sec time (Pentium IV, 3.2 GHz, 3 GB RAM). Post the SVD, the spaces are loaded into RAM and calculation of text similarities take only fractions of millisecs. The results reported in this paper have been done on clients running in Paris, while the LSA-server had been placed in Würzburg/Germany.

2.4 Hypotheses

We have two sets of hypothesis : the first set corresponds to the characteristic that an ideal memory space should have to get the best performance in order to solve a categorization task. The second set of hypthesis corresponds to the materials characteristics in order to apply the similarity comparison.

2.4.1 Caracteristics of semantic spaces

Our first set of hypothesis concerns the « experience » effect that the memory space build would have. We contrast two cases. In the first case, we build a semantic space corresponding to a big semantic memory where all types of knowledge are mixed . In the second case, we construct different semantic spaces which correspond to specific types of knowledge. The first will be called « the Big box method »and the second case will be called the « small box » method. We test which case has the best perfomance in a categarisation task.

2.4.2 Caracteristics of the material used to applied comparisons

Our second set of hypotheses concerns the characteristics of the mental representation corresponding to each category between which we have to choose. The idea is that, in a situation of a categorization task, a subject should have a proper mental representation of the different categories he/she have to choose between. Let us consider the case, that someone gives me a critic of a movie. To tell something about the jugment given by the critic, I must know something about how people make their judgment about a good movie or a bad movie. In other words, I must have a proper mental representation of « what is a good critic of a movie » and « what is a bad critic of a movie ». To follow this idea, if a subject has a proper mental representation of the categories that he/she has to choose between, the second step should be to compare the similarity between the incoming object and the mental representation for each category then he has to choose between them the answers of the category is the one that is the most similar to the object. For each category, we create a vector corresponding to all the documents of this category. For example, for the category « good movie », we compute the corresponding vector to the whole document « good movie ». We assume that this vector corresponds to the intention of the category « good movie ». We assume also that this vector corresponds to the mental representation of the categorys intention « good movie ». Following this idea, in order to know if a critic corresponds to the category « good movie », we will compare it with the vector corresponding the category « good movie ». These vectors should be like targets : the more specific they are, the better they are, then more accurate the results will be. We call : target files or target vectors.

3 Building and Training of Spaces

As soon as we had obtained the training data from Deft, the top priority was to build the spaces necessary for classification. Table 1, shows the distribution of documents among types and

judgments in the training data. In the table, the rows represent the types, while columns stand for judgments in the types. In particular, there is no judgment 2 in type Débats.

	0	1	2	Total
aVoiraLire	309	615	1150	2074
Débats	10400	6899	0	17299
Jeuxvidéo	497	1166	874	2537
Relecture	227	278	376	881
				22791 (Total)

Table 1. Distribution of Documents through Types and Judgments.

From Table 1, it is clear that the documents are not evenly distributed among types and judgments. This gives rise to two questions: i) If all the documents are used for building spaces, would this result in unbalanced spaces? ii) If we balance the documents to build a balanced space, would the left out documents cause information loss? Apart from those, there is another consideration about the space: Should we use a big space (big box) for all the types, or one small space (small box) per type? After trying different configurations, we decided to use 3 different methods to build spaces for the final tests:

1 Big box space balanced by number.
2 Small box spaces balance by number.
3 Small box spaces with all the documents.

We can see, there is one big box method, while there are two small box methods. During the unofficial tests, though, the big box method does not perform as well as small box methods, the overloaded LSA server prevented us from finishing the official tests with small box spaces which is balanced by number before the deadline. Hence we included the big box results in the official test results. In the following subsections, the details about those spaces are described.

At this point we have successfully extracted every document from XML files, and organized them in hierarchic directories: everything is ready for space construction.

3.1 Big box space balanced by number

In the purpose of producing a balanced LSA matrix, we decided to balance the number of documents for building the space. In Table 2 the balanced numbers for Documents are shown. We choose 300 dimensions for every judgment, because this is the closest number to the least numbered judgment, i.e. type Relecture, judgment 0 and 1. There are at least 10 documents that are not selected from each judgment in every type, even if the documents in the judgment are not enough to extract 300 dimensions, and they are for unofficial testing purpose. This means we have 120 documents in total from all the types and judgments. In the case of type Jeuxvidéo, there is no document in judgment 2, thus 450 documents are chosen for each of judgment 0 and 1 to make a total number of 900 for the type. The same for the unofficial test documents: 15 documents are chosen from each type of Juexvidéo to make a total number of 30 documents for the type.

	0	1	2	total
aVoiraLire	299	300	300	899
Débats	450	450	0	900
Jeuxvidéo	300	300	300	900

Relecture	217	268	300	785
Bigbox				3484

Table 2. Document numbers for balanced by number space.

3.2 Small box spaces with all the documents

The one big box for all the types may works well for distinguishing types, which is already known, whilst it may not work well for judgments inside types because of the noise from other types inside the space. A solution is to split the big box space into four smaller spaces, each corresponding to a type. In this small box method, no balancing is used, which means all the training documents are used to build the spaces. This guarantees the spaces are trained by all the information available. In Table 3, we can see the contents of these spaces. Thanks to their unbalanced feature, the table can be directly derived from Table 1.

	0	1	2	Total Documents
aVoiraLire space	309	615	1150	2074
Débats space	10400	6899	0	17299
Jeuxvidéo space	497	1166	874	2537
Relecture space	227	278	376	881

Table 3, small box spaces with all the documents.

3.3 Small box spaces balanced by number

We have already made an argument for small box above in subsection 3.2. However, with all the documents included for a space, there may occur unbalance inside the space. For example, in Table 3, the aVoiraLire space includes 2074 documents, 15% of which are from judgment 0, 30% are from judgment 1, and the rest i.e. 55% are from judgment 2. Being afraid that the unbalanced spaces may influence test results, the small box spaces are balanced by number in this method. Table 4, shows the contents of every space.

	0	1	2	Total Documents
aVoiraLire space	309	309	309	927
Débats space	6899	6899	0	13798
Jeuxvidéo space	497	497	497	1491
Relecture space	227	227	227	681

Table 4, Small box spaces balanced by number

3.4 Other spaces

There are some other methods we have tested but have dumped for different reasons.

1 Small box spaces for judgments. There are three spaces which contain documents from judgment 0, 1 and 2 respectively. The spaces do not differentiate between types. The unofficial tests gave very poor results for this method, and we dumped it. The argument is that even if the spaces seem to be organized by judgment similarity, the words for judgment vary in different types, which in turn diminishes the similarity. Further more, documents from different types bring noise to each other, thus make judgment more inaccurate.

2 Small box spaces balanced by percentage. This method is the same as the method in 3.2 apart from we balance the documents by percentage. For example we take 90 percent of documents from each judgment to make the spaces. This seems we have "balanced" the documents, but the unbalanced feature of the space will not be solved by this improper balancing method, besides there some information loss occurs, due to the not included documents. In the other hand, we can always produce the method 3.2 by increasing the percentage to 100 percent. Thus when the percentage increases, the result will be similar to the 3.2 method.

4 Final Tests and arguments

With the above spaces, we are almost ready to do the tests. We performed three different tests, each of which corresponded to each of the above spaces. But before that, for every test we had to prepare target files and test files. Test files were the same for every test, while the target files differed from one another. In Table 5, the test files' contents are shown.

Type	Test Documents
aVoiraLire	1388
Débats	11534
JeuxVidéo	1695
Relecture	603

Table 5, Official test documents.

If a space, its target files and test files are given, we are able to perform the test. Below code explains the algorithm for the test:

```
Declarations:
        target-file0; //judgment 0
        target-file1; //judgment 1
        target-file2; //judgment 2
        test-file; //test file
        space;   //lsa space
Program:
        result0 = cosine(space, target-file0, test-file);
        result1 = cosine(space, target-file1, test-file);
        result2 = cosine(space, target-file2, test-file);
        result = max(result0, result1, result2);
        if ( result == result0)
                then judgment = 0;
        else if (result == result1)
                then judgment = 1;
        else judgment = 2;
        return judgment;
```

Let us explain the code. The declaration part declares target-filetarget-files, test-file and space. These information could be passed to this part of code as parameters. Then the program compares the target-file and the test-file. The comparison is done by computing the cosine between two vectors of these two files in the given space (see more details in section 2). And the last part of the code is to find out which result is the maximum one, and to which target-file it corresponds to. This means when the cosine between a target-file and the test-file is maximum, the test-file is close to this target-file, rather than two others. From this, we say the test-file belongs to the judgment to which the target-file belongs. There some points we would like to make clear. i) This piece of code is only for one test-file, for multiple test files it should be called by another program in a loop structure. Of course, in a real world implementation, this code could be embedded into a program with more efficiency in mind. ii) In the code there are three target files, but in the case of Débats , there should be only two target files.

We have explained test files and spaces but not yet target files. Target files are the one against which we compare our test files. In general it contains all the documents from one judgment which are used to build spaces. So with the change of space, the target files change. In the following subsections, we will talk about test target files, and test results.

4.1 Test for big box space balanced by number

In this test, there are 11 target files every one of which corresponds to a judgment. We can use Table 2 to explain the contents of a target file. For example, a target file for type relecture, judgment 0, contains 217 documents. We don't have to change spaces, since there is only one space.

The results are shown below:

Corpus avoir-alire (1), exécution 1, F-score pondéré :

Précision: 0.459110186683814 Rappel: 0.49455465710474

F-score pondéré: 0.476173746841085

Corpus des jeux vidéo (2), exécution 1, F-score pondéré :

Précision: 0.647385184622607 Rappel: 0.633138889933088

F-score pondéré: 0.64018278968055

Corpus des relectures (3), exécution 1, F-score pondéré :

Précision: 0.397253652191251 Rappel: 0.399309105766007

F-score pondéré: 0.398278727028512

Corpus des débats (4), exécution 1, F-score pondéré :

Précision: 0.577640356467526 Rappel: 0.575419817485701

F-score pondéré: 0.576527948843027

4.2 Test for small box space with all the documents

Here we use 4 small spaces, and 11 target files. Target files can be explained using Table 3.
the results are:

Corpus avoir-alire (1), exécution 2, F-score pondéré :
Précision: 0.610107317646253 Rappel: 0.58790185104376
F-score pondéré: 0.598798791785183

Corpus des jeux vidéo (2), exécution 2, F-score pondéré :
Précision: 0.740923815003921 Rappel: 0.663222700360692
F-score pondéré: 0.699923388295196

Corpus des relectures (3), exécution 2, F-score pondéré :
Précision: 0.503115560359155 Rappel: 0.510760891580065
F-score pondéré: 0.506909400421208

Corpus des débats (4), exécution 2, F-score pondéré :
Précision: 0.679694375399586 Rappel: 0.683170682019693
F-score pondéré: 0.681428095142408

4.3 Test for small box spaces balanced by number

We use four small spaces and 11 target files. Target files can be explained by Table 4. Even if the result of this test is not officially submitted, we believe they are worth mentioning here.
there results are:

Corpus avoir-alire (1), exécution 3, F-score pondéré :
Précision: 0.386077238255826 Rappel: 0.408823220435605
F-score pondéré: 0.397124792556954

Corpus des jeux vidéo (2), exécution 3, F-score pondéré :
Précision: 0.644021148754158 Rappel: 0.591851261951918
F-score pondéré: 0.616835081537432

Corpus des relectures (3), exécution 3, F-score pondéré :
Précision: 0.453810019726474 Rappel: 0.462140584841882
F-score pondéré: 0.457937419064932

Corpus des débats (4), exécution 3, F-score pondéré :
Précision: 0.670371869281329 Rappel: 0.67373015608171
F-score pondéré: 0.672046817281907

4.4 Discussion

From the results we can see, the 4.2 test has the best result. 4.1 has the worst result. and 4.3 is a little bit better than 4.1 but not so good as 4.2. These results show the better performance in the categorization task when similarity is computed from a specific memory space than when it is computed from a global memory space. Some similar results have been found concerning relative judgement of importance of sentences in comprehension of narratives (Denhière, Hoareau, Jhean-Larose, Lehnard, Baïer, Bellissens,2007).

5. Conclusion

We have implemented two hypothesis about the categorization activity. We have compared two methods called “Big box” versus “Small Box” that correspond to two hypothesis about the type of

knowledge organization of semantic memory. The results are better with - the “Small box” method than with the “Big Box” method, even if the resulting semantic spaces are constructed of a small amount of data and that LSA is a statistical method.

6. Literature

BERRY, M. W, DUMAIS, S. T, AND O'BRIEN, G. W. (1995). Using linear algebra for intelligent information retrieval. *SIAM Review, 37(4)*, 573-595.

CACCAMISE, D., FRANZKE, M., ECKHOFF, A., KINTSCH, E., & KINTSCH, W. (2007). Guided Practise in Technology-Based Summary Writing. In D. S. MCNAMARA (Ed.), Reading Strategies (375-396). Mahwah, N. J., Erlbaum.

DEERWESTER, S., DUMAIS, S. T., FURNAS, G. W., LANDAUER, T. K., & HARSHMAN, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society For Information Science, 41*, 391-407.

DENHIÈRE, G., LEMAIRE, B., BELLISSENS, C. & JHEAN-LAROSE, S. (2005). Psychologie cognitive et compréhension de texte : une démarche théorique et expérimentales. In S. Porhiel & D. Kintgler (Eds), L’unité texte (pp. 74-95) Pleyben : Perspectives.

DENHIÈRE, G., LEMAIRE, B., BELLISSENS, C. & JHEAN-LAROSE, S. (2007). A semantic space for modeling a child semantic memory. In T. LANDAUER, D.S. MCNAMARA, S. DENNIS, & W. KINTSCH (Eds), *Handbook of Latent Semantic Analysis (pp. 143-167).* Hillsdale, N.J. : Lawrence Erlbaum Associates.

DENHIERE, G., HOAREAU, Y. V., JEAN-LHAROSE, S., LENHARD,W.,BAIER,H., BELLISSENS, C. (2007). Human Hierarchization of Semantic Information in Narratives and Latent Semantic Analysis. *Proceedings of the First European Workshop on Latent Semantic Analysis in Technology Enhanced Learning,* 15-17.

DUMAIS, S. T. (1991). Improving the retrieval of information from external resources. *Behavior Research Methods, Instruments, & Computers, 23*, 229-236.

FOLTZ, P. W., KINTSCH, W.,& LANDAUER, T. K. (1998). The measurement of textual Coherence with latent Semantic Analysis. *Discourse Processes, 25*, 285-307.

GRAESSER, A., WIEMER-HASTINGS, K., WIEMER-HASTINGS, P., KREUZ, R., & THE TUTORING RESEARCH GROUP (1999). AutoTutor: A simulation of a human tutor. *Journal of Cognitive Systems Research, 1*, 35-51.

KONTOSTATHIS, A., & POTTENGER, W.M. (2002). Detecting patterns in the LSI term-term matrix. *Workshop on the Foundation of Data Mining and Discovery, IEEE International Conference on Data Mining*.

LANCZOS, C. (1950). An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators [Online]. Available: http://www.cs.duke.edu/courses/fall06/cps258/references/Krylov-space/Lanczos-original.pdf, date of retrieval: 10.01.2007). *Journal of Research of the National Bureau of Standards, 48*, 255-282.

LANDAUER, T. K., & DUMAIS, S. T. (1997). A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review, 104*, 211-240.

LANDAUER, T. K., FOLTZ, P. W., & LAHAM, D. (1998). Introduction to Latent Semantic Analysis. *Discourse Processes, 25*, 259-284.

LANDAUER, T. K., LAHAM, D., REHDER, B., & SCHREINER, M. E., (1997). How well can passage meaning be derived without using word order? A comparison of Latent Semantic Analysis and humans. In M. G. SHAFTO, & P. LANGLEY (Eds.), *Proceedings of the 19th annual meeting of the Cognitive Science Society (pp. 412-417)*. Mawhwah, NJ: Erlbaum.

LEMAIRE, B., & DENHIÈRE, G. (2004) Incremental Construction of an Associative Network from a Corpus. In K. FORBUS, D. GENTNER, & T. REGIER (Eds.), *Proceedings 26th Annual Meeting of the Cognitive Science Society (825-830)*, Chicago.

LEMAIRE, B., DENHIÈRE, G., BELLISSENS, C. & JHEAN-LAROSE, S. (2066). A model and a computer program for simulating text comprehension. *Behavior Research Methods,* 38 (4), 628-637.

LENHARD, W. (in submission). Bridging the Gap to Natural Language: Latent Semantic Analysis as a Tool for the Development of Intelligent Tutoring Systems. LENHARD, W., BAIER, H. SCHNEIDER, W., & HOFFMANN, J. (2006). Forschungsprojekt "Förderung des

Textverständnisses": LSA-Modul [Research Project „Enhancing text comprehension“, LSA-Module]. accessible under: http://www.summa.psychologie.uni-wuerzburg.de/summa/coa/login/ (last accessed April 2007).

LENHARD, W., BAIER, H., HOFFMANN, J., & SCHNEIDER, W. (in press). Automatische Bewertung offener Antwortformate mittels Latenter Semantischer Analyse [Automatic Scoring of Open Text Responses via Latent Semantic Analysis]. *Diagnostica*.

LENHARD, W., BAIER, H., HOFFMANN, J., SCHNEIDER, W., & LENHARD, A. (2007). Training of summarisation skills via the use of content based feedback. *Proceedings of the First European Workshop on Latent Semantic Analysis in Technology Enhanced Learning,* 26-27.

Martin, D., & Berry, M. (2007). Mathematical Foundations Behind Latent Semantic Analysis. In T. K. Landauer, D. S. McNamara, S. Dennis, & W. Kintsch (Eds.), *The Handbook of Latent Semantic Analysis*. Mahwah, NJ: Erlbaum.

NAKOV, P. (2000). Getting Better Results With Latent Semantic Indexing. *Proceedings of the Students Presentations at the European Summer School in Logic Language and Information (ESSLLI'00)*, 156-166.

NAKOV, P., POPOVA, A., & MATEEV P. (2001). Weight functions impact on LSA performance. *Proceedings of the EuroConference Recent Advances in Natural Language Processing, RANLP'01*, 187-193.

REHDER, B., SCHREINER, M. E., WOLFE, M. B., LAHAM, D., LANDAUER, T. K., & KINTSCH, W. (1998). Using Latent Semantic Analysis to assess knowledge: Some technical considerations. *Discourse Processes, 25*, 337-354.

TISSERAND, D., JHEAN-LAROSE, S., DENHIÈRE, G. (2007). Eye movement analysis and Latent Semantic Analysis on a comprehension and recall activity. *Proceedings of the First European Workshop on Latent Semantic Analysis in Technology Enhanced Learning,* 36-37.

WADE-STEIN, D., & KINTSCH, E. (2004). Summary Street: Interactive computer support for writing. *Cognition and Instruction, 22*, 333-362.

WIEMER-HASTINGS, P. (1999). How latent is Latent Semantic Analysis? *Proceedings of the Sixteenth International Joint Congress on Artificial Intelligence (pp. 932- 937).* San Francisco: Morgan Kaufmann.

WILD, F., STAHL, CH., STERMSEK, G., & NEUMANN, G. (2005). Parameters Driving Effectiveness of Automated Essay Scoring with LSA. *Proceedings of the 9th International Computer Assisted Assessment Conference,* 485-494.

WOLFE, M. B., SCHREINER, M. E., REHDER, B., LAHAM, D., FOLTZ, P. W., KINTSCH, W., & LANDAUER, T. K. (1998). Learning from text: Matching readers and text by Latent Semantic Analysis. *Discourse Processes, 25*, 309-336.