# On the Use of Topic Models for Word Completion

Elisabeth Wolf[1], Shankar Vembu[1], and Tristan Miller[2,⋆]

[1] German Research Center for Artificial Intelligence
Erwin-Schroedinger-Strasse 57, 67663 Kaiserslautern, Germany
{wolf, vembu}@dfki.uni-kl.de
[2] The Socialist Party of Great Britain
52 Clapham High Street, London SW4 7UN, United Kingdom
tristan.miller@worldsocialism.org

**Abstract.** We investigate the use of topic models, such as probabilistic latent semantic analysis (PLSA) and latent Dirichlet allocation (LDA), for word completion tasks. The advantage of using these models for such an application is twofold. On the one hand, they allow us to exploit semantic or contextual information when predicting candidate words for completion. On the other hand, these probabilistic models have been found to outperform classical latent semantic analysis (LSA) for modeling text documents. We describe a word completion algorithm that takes into account the semantic context of the word being typed. We also present evaluation metrics to compare different models being used in our study. Our experiments validate our hypothesis of using probabilistic models for semantic analysis of text documents and their application in word completion tasks.

## 1 Introduction

Word completion is the task of predicting and automatically completing words that the user is in the process of typing. Such tools can prevent misspellings, help develop writing skills, and accelerate typing speed by saving keystrokes. (The last benefit is particularly important for users of keyboardless devices, such as mobile phones and PDAs, as well as for users with physical disabilities.) During typing, the user is offered a prediction list of words beginning with the letters, or word prefix, thus far typed. If the intended word is in the prediction list, the user can select it with a single keypress; otherwise, he continues typing until the word appears in the list or until he types the complete word.

The job of the word completion algorithm is to determine which words appear in the prediction list, the idea being to maximise the probability of presenting the user with the correct word. The earliest word completion algorithms [1] used simple statistical methods, such as word or word-pair frequencies, to rank words

⋆ Part of the research described in this paper was carried out while this author was at the German Research Center for Artificial Intelligence.

in the prediction list. The frequencies are derived from a corpus of written text, though some systems [2] dynamically update the frequency table to adapt to the user's writing style. More advanced systems [3] incorporate syntactic data, such as part-of-speech tags and grammar rules, to avoid suggesting words which are grammatically incorrect in the given context. However, even systems that combine statistical and syntactic information can suggest words that are *semantically* inappropriate. For instance, the writer of an essay on music who begins typing *Mende...* is far more likely to intend the completion to be *Mendelssohn* than *Mendel* or *Mendeleyev*, even though all three are proper nouns that may be equally statistically likely (in a unigram or a bigram model, at least).

In order to avoid suggesting semantically inappropriate words, several approaches were proposed in which semantic knowledge is incorporated into the completion task. An early attempt at incorporating semantic information was proposed by Kozima and Ito [4]. They deal with a scene-based model that uses local semantic information of each scene—i.e., a text fragment which displays a semantic unit. However, they predict words based on context-sensitive word distances, because there is no training corpus segmented into scenes to derive probabilities of the occurrence of a word given a text fragment. Other recent attempts (e.g., [5]) require language-specific tools such as WordNet [6], and many operate only on words of a particular part of speech.

All of these approaches have shown an improvement of the word completion task in predicting semantically more appropriate words. But none of these explicitly model the semantics of text documents resulting in disambiguation of polysems and synonyms, which is possible using models like latent semantic analysis (LSA) [7]. In our recent work [8], we demonstrated the advantages of exploiting the semantic context of words that have been typed for predicting a list of candidate words for completing the current word using LSA. In this paper, we investigate the application of topic models—namely, probabilistic latent semantic analysis (PLSA) [9] and latent Dirichlet allocation (LDA) [10]—to model the semantics of text documents. In recent years, these models have been gaining widespread interest as semantic models not only of text collections but also in other domains like images [11,12]. We make empirical comparisons of these models for word completion tasks with LSA as the baseline model.

The paper is organised as follows: We begin with a brief description of topic models that we intend to use in our experiments. We then present a semantic-based word completion algorithm in Section 3 and give a complexity analysis. In the following section, we describe the details of our simulator for word completion and present evaluation metrics that will be used for the comparison of the various topic models. Section 5 describes our experimental work, and is followed by conclusions and pointers to future work.

## 2 Topic Models

LSA has been in use for a long time for the automatic indexing and retrieval of text documents. It is based on the singular value decomposition (SVD) of the term–document matrix $X$ giving rise to two orthogonal matrices $U$ and $V$, and a diagonal matrix $\Sigma$, such that $X = U\Sigma V^T$. The elements of $\Sigma$ are called singular values and the columns of $U$ and $V$ are called left and right singular vectors, respectively. A reduced-rank approximation of $X$ is obtained by discarding all but the highest $K$ singular values in $\Sigma$. The resulting matrices define the so-called *latent semantic space* in which common information retrieval operations such as comparison of two terms, two documents, or a term and a document can be performed.

PLSA is the probabilistic version of LSA and it defines a generative model for statistical modeling of discrete and count data of which text collections are an example. PLSA assumes the existence of a latent variable $z_k \in \{z_1, \ldots, z_K\}$, where $K$ is the number of topics, for each word (or observation) in a document. The data generation process is described in three steps: a document $d_i$ is selected with probability $p(d_i)$; a latent class variable $z_k$, also referred to as the topic variable, is selected with probability $p(z_k|d_i)$; a word $w_j$ is finally generated with probability $p(w_j|z_k)$. The probability of an observation pair $(d_i, w_j)$ is given as

$$p(d_i, w_j) = p(d_i) \sum_{k=1}^{K} p(w_j|z_k)p(z_k|d_i) \ .$$

The model parameters are estimated using the expectation-maximisation (EM) [13] algorithm. If we assume a corpus of $M$ documents and a vocabulary of $N$ words, the parameters of a $K$-topic PLSA model are $K$ multinomial distributions of size $M$ and $N$ mixtures over the $K$ hidden topics, thereby making the total number of parameters to be $KN + KM$. The linear dependence of the number of parameters on the size of the corpus results in overfitting and a tempered version of EM was proposed by Hofmann [9] to mitigate this problem. The inference step involves estimating the distribution of the topics given a new document—i.e., $p(z_k|d_{new})$—by fixing the $p(w_j|z_k)$ parameters. This step, also called *folding in* [7], projects new, unseen documents into the latent semantic space.

LDA is a three-level hierarchical Bayesian model in which each document is modeled as a mixture of an underlying set of topics, very much similar in a sense to PLSA. But the drawbacks of PLSA, such as its linear dependence on the number of documents for parameter estimation and its inability to assign probability to previously unseen documents, are mitigated in the LDA model [10]. The data generation proceeds as follows: a Dirichlet parameterised by $\alpha$ is sampled to yield $\theta$; for each of the $N$ words, a topic $z_n$ is sampled from a multinomial parameterised by $\theta$ and a word $w_n$ is chosen with probabibilty $p(w_n|z_n, \beta)$, which again is a multinomial conditioned on the topic $z_n$. The model parameters are given by $\alpha$ and $\beta$. The probability of a corpus $D$ consisting of $M$ documents having $N$ words in each of them is given as

$$p(D|\alpha, \beta) = \prod_{m=1}^{M} \int p(\theta_m|\alpha) \left( \prod_{n=1}^{N_m} \Sigma_{z_{mn}} p(z_{mn}|\theta_m) p(w_{mn}|z_{mn}, \beta) \right) d\theta_m \ .$$

The number of parameters in a $K$-topic LDA model is $K + KM$—i.e., the Dirichlet parameter $\alpha \in \mathbb{R}^K$ and the $K$ multinomial word distributions. Therefore, unlike PLSA, parameter estimation in LDA is not dependent on the number of documents $N$. The inferential quantity of interest is the distribution of the topics given a new document $p(\theta, z|\mathbf{w}, \alpha, \beta)$ and is estimated using approximate inference techniques for graphical models [14].

## 3 Semantic Word Completion

### 3.1 Algorithm

The first step is to build semantic models of the text corpus using LSA, PLSA and LDA. Ideally the training corpus should be large enough to contain any word the user is likely to type. Once the models are built, pairs of term or document vectors can be compared via the cosine coefficient, yielding a "semantic similarity" score in the range $[-1, 1]$. Assume the user is in the process of typing a word $w$ with prefix pre$(w)$. We define the *context* $C = \langle c_1, c_2, \ldots, c_{\ell-1}, c_\ell \rangle$ as the sequence of up to $\ell$ words immediately preceding $w$ in the document. We refer to $\ell$ as the *context length*, though near the beginning of the document the actual length of the context, $|C|$, may be less than $\ell$.

A *candidate word* $t \in T \subseteq V$, where $V$ is the vocabulary of size $N$, is any word whose prefix is the same as that of $w$—i.e., $T = \{t | t \in V \wedge \text{pre}(t) = \text{pre}(w)\}$. The best candidates comprise the *prediction list* $P \subseteq T$, which the user (or system) caps at a maximum length of $p$. Again, it is possible that $|P| < p$ if there are fewer than $p$ known words with the given prefix. We propose a method called *sum of similarities* (SOS) to populate $P$, in which we compare the candidates to each word in the context individually. The similarity score for the context is the sum of similarity scores of each word in the context and is given as

$$\text{sim}(t, C) = \sum_{i=1}^{|C|} \cos(t, c_i) \, .$$

We compute the similarity scores for each possible $t$, and populate the prediction list $P$ with $p$ high-scoring candidates.

### 3.2 Complexity Analysis

The application of topic models to word completion involves two steps: creating models (or parameter estimation) of LSA, PLSA and LDA; and simulation of word completion using the SOS algorithm. The input to our system is an $N \times M$ term–document matrix and let the desired number of topics be $K$. The time complexity of building an LSA model depends on the SVD of matrices. Efficient algorithms to perform SVD can be found in the literature. For example, Brand [15] introduced an algorithm that performs a reduced-rank SVD of a matrix in $\mathcal{O}(N \cdot M \cdot K)$ time, which is linear in the number of inputs and outputs. The estimation of model parameters in PLSA and LDA is performed using the EM algorithm, and therefore the time complexity is dependent on the number of operations per EM iteration and also on the number of iterations

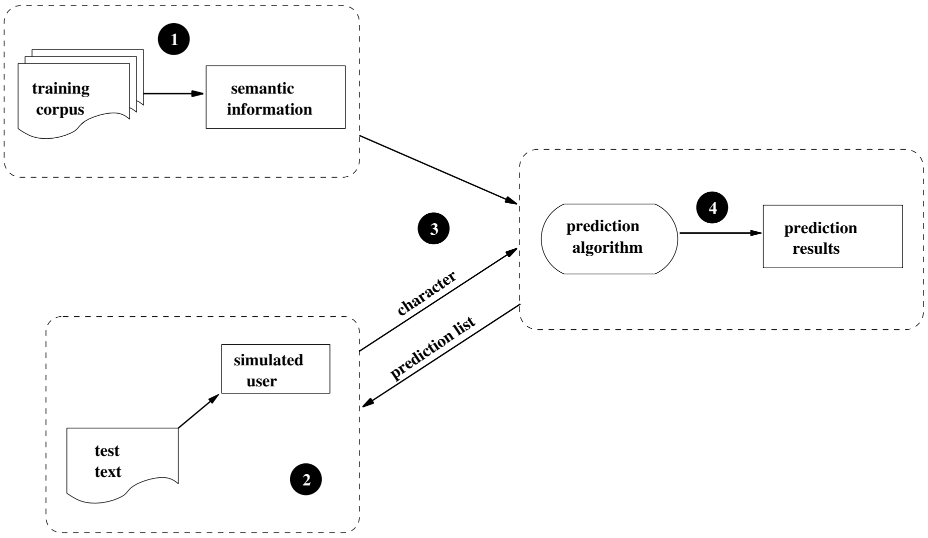

**Fig. 1.** Simulation architecture

that are required for convergence. Each EM iteration in PLSA requires $\mathcal{O}(R \cdot K)$ operations, where $R$ is the number of distinct observation pairs $(d_i, w_j)$—i.e., $N \cdot M$ times the degree of sparseness of the term–document matrix. The number of iterations typically falls in the range 20–50 [9]. Each iteration in the varational E-step in LDA requires $\mathcal{O}((N+1) \cdot K)$ operations, with the number of iterations for a single document roughly equalling the number of words $M$ in the document [10]. We thus observe that the time complexity of all the three models is linear in the individual inputs. In the SOS algorithm, we perform a term–term comparison i.e., a cosine operation for each word in the context, and therefore the number of comparisons is in the order of $\mathcal{O}(\ell \cdot |T|)$, where $\ell$ is the context length. Note that the SOS algorithm does not require the inference step of PLSA and LDA, since it performs only a term–term comparison using the $K$ multinomial word distributions that are readily available after the parameter estimation step.

## 4 Test Bench

### 4.1 Simulation

We designed and implemented a simulator in order to evaluate the performance of our word completion algorithm. The simulator (illustrated in Figure 1) consists of three major components. The first component (❶) covers all necessary preprocessing steps and trains LSA, PLSA, LDA models in order to extract semantic information. In the second component (❷), a simulated user is integrated, who interacts (❸) with the prediction component (❹).

The simulated user types in the words of a test document by passing character after character to the prediction component, and gets $p$ candidates presented in the prediction list in return. The population of the prediction list is dependent on the context and the semantic model being used. In order to choose a limited number of candidates for the prediction list, the prediction algorithm calculates the semantic similarity of each candidate and selects the $p$ most appropriate words. Afterwards one of the three following cases can occur. In the explanation below, $P$ is the prediction list and $w$ is the typed word with prefix $\text{pre}(w)$ that has to be completed.

**Case 1:** $w \in P$
The word appears in the prediction list. It is selected and the system proceeds with the first character of the next word.

**Case 2:** $w \notin P, |\text{pre}(w)| < |w|$
The intended word does not appear in the prediction list and is not typed totally thus far. The word prefix is expanded by the next character of the current word and passed to the prediction algorithm.

**Case 3:** $w \notin P, |\text{pre}(w)| = |w|$
The intended word could not be completed before it was completely typed.

During the whole experimental process, detailed information is stored for further analysis. The simulation terminates after the whole test text has been processed.

## 4.2 Evaluation Metrics

Performance of the system is assessed with the following three metrics:

**Keystroke savings,** the most important metric, is the percentage of keystrokes that the user saves by using the word completion utility:

$$\text{KS} = 100 - \frac{100}{|W|} \cdot \sum_{w \in W} \frac{s_w + 1}{\text{len}(w)},$$

where $|W|$ is the number of words in the test set of documents, $s_w$ is the number of keystrokes used to type a given word $w$, $+1$ is the one additional keystroke to choose the appropriate word in the prediction list, and $\text{len}(w)$ is the number of characters in $w$—i.e., the number of keystrokes that would have had to be typed without the word completion utility.

**Hit rate** refers to the percentage of keystrokes after which the intended word appears in the prediction list:

$$\text{HR} = 100 \cdot \left[ \sum_{w \in W} \text{in}(w) \right] \div \sum_{w \in W} s_w ,$$

where

$$\text{in}(w) = \begin{cases} 1 \text{ if } w \in P \text{ after typing } s_w \text{ characters;} \\ 0 \text{ otherwise.} \end{cases}$$

**Keystrokes until prediction** is the mean number of keystrokes until the intended word appears in the prediction list or is completely typed:

$$\text{KUP} = \frac{1}{|W|} \cdot \sum_{w \in W} s_w \,.$$

## 5 Experiments and Results

### 5.1 Data Sets

We used Reuters-21578 [16] corpus with the ModApte splitting scheme for our experiments. The ModApte split results in a corpus of 9603 training documents and 3299 test documents. Although the original corpus has 135 topics, the ModApte split yields only 90 topics for which there is at least one training and one test document. We performed standard preprocessing techniques of stop word removal and stemming. We also removed all words that occurred in less than three documents and in more than 90% of the training documents. These operations resulted in a $5605 \times 9603$ term–document matrix which was used to build our models using LSA, PLSA, and LDA. Since PLSA and LDA operate on discrete or count data, we used the word counts instead of the standard tf–idf representation of documents.

### 5.2 Training Phase

We trained LSA, PLSA, and LDA for different values of $K$, the number of topics, on the training set consisting of 9603 documents. We trained the PLSA model using the tempered version of EM as described by Hofmann [9] to avoid the possibility of overfitting. The LDA model was trained using variational EM as described in Blei's paper [10]. The Dirichlet parameter $\alpha$ was set to an initial value of 0.5 and was allowed to be iteratively estimated along with the topic distributions. We used the same stopping criteria of 200 maximum iterations and 0.0001% change in expected log likelihood (whichever of the two occurs first) for training PLSA and LDA models. Training an LSA model simply entails performing an SVD on the term–document matrix and as such there were no free parameters to fine tune.

### 5.3 Simulation Results and Analysis

Simulating the word completion algorithm SOS is a computationally expensive operation, since a term–term comparison has to be made for each word in the context. We therefore could not use the entire test set of documents with all the words for our experiments. Instead, we sampled the test corpus in such a way so as to include two documents from each of the 90 topics. This resulted in 120 documents with 12 942 words that was finally used in our simulation. We note that in the Reuters corpus, a single document might be assigned to multiple topics, and therefore we have 120 instead of 180 documents.

Simulation results for $K = 25$ topics are shown in Figure 2. The figures depict the performance of LSA, PLSA, and LDA for the three metrics defined in §4.2 as a function of context length. It is evident from these graphs that the probabilistic models PLSA and LDA outperform LSA, and the best performance is achieved by LDA for all the evaluated metrics. For instance, for a context length of 14 words, LDA outperforms LSA by 8.19%, 9.94%, and 9.06% in terms of keystroke savings, hit rate, and keystrokes until prediction, respectively. An interesting observation is the dependence of these metrics on context length. In contrast to LSA, we note that PLSA and LDA show a significant increase in keystroke savings and hit rate with context length. The same holds true for keystrokes until prediction, which decreases with context length. These observations suggest that PLSA and LDA are able to model semantics or contextual information in a much better way when compared to LSA. For all three models, we observed that the performance of the system did not improve continuously for higher values of context length. For instance, we see that the keystroke savings for LSA improve until the context length is 6, but thereafter the performance goes down. The same holds true for the other metrics. For PLSA and LDA, the increase in keystroke savings was not significant for larger context lengths. This might suggest that incorporating too much semantic or contextual information for word prediction does not help; for a given application the appropriate context length could be determined by the available computational resources and with reference to these results. We also note from the complexity analysis in §3.2, that the response time of a word completion utility is linearly dependent on the context length, and therefore having large context lengths might slow down the overall response of the system.

We proceed with our analysis of results for different values of $K$, the number of topics. Simulation results are shown in Figure 3 for $K = 50$ and $K = 75$. We observe that there is indeed a performance gain as the number of topics increases. An interesting observation is the way PLSA behaves for higher numbers of topics. For $K = 25$, PLSA fares better than LSA at every context length and evaluation metric. But, for $K = 50$, we see that for lower values of context lengths, LSA performs better than PLSA, and this tendency becomes marked when we increase the number of topics to 75. This might be the result of overfitting due to an increase in the number of parameters. Interestingly, LDA did not suffer from any such problems and it seems to fare better with larger context lengths. For instance, with a context length of 14, LDA with $K = 75$ performs better than with $K = 25$ by 4.34%, 6.04%, and 5.63% in terms of keystroke savings, hit rate, and keystrokes until prediction, respectively. It is not computationally feasible to experiment with many values of $K$, and therefore we are not able to report results for a wide range of values of $K$. Nonetheless, as described above, we are able to draw some important conclusions regarding the behaviour of these models for different number of topics. The number of topics is indeed a bottleneck parameter, and the best value for it is dependent on the available computational resources.

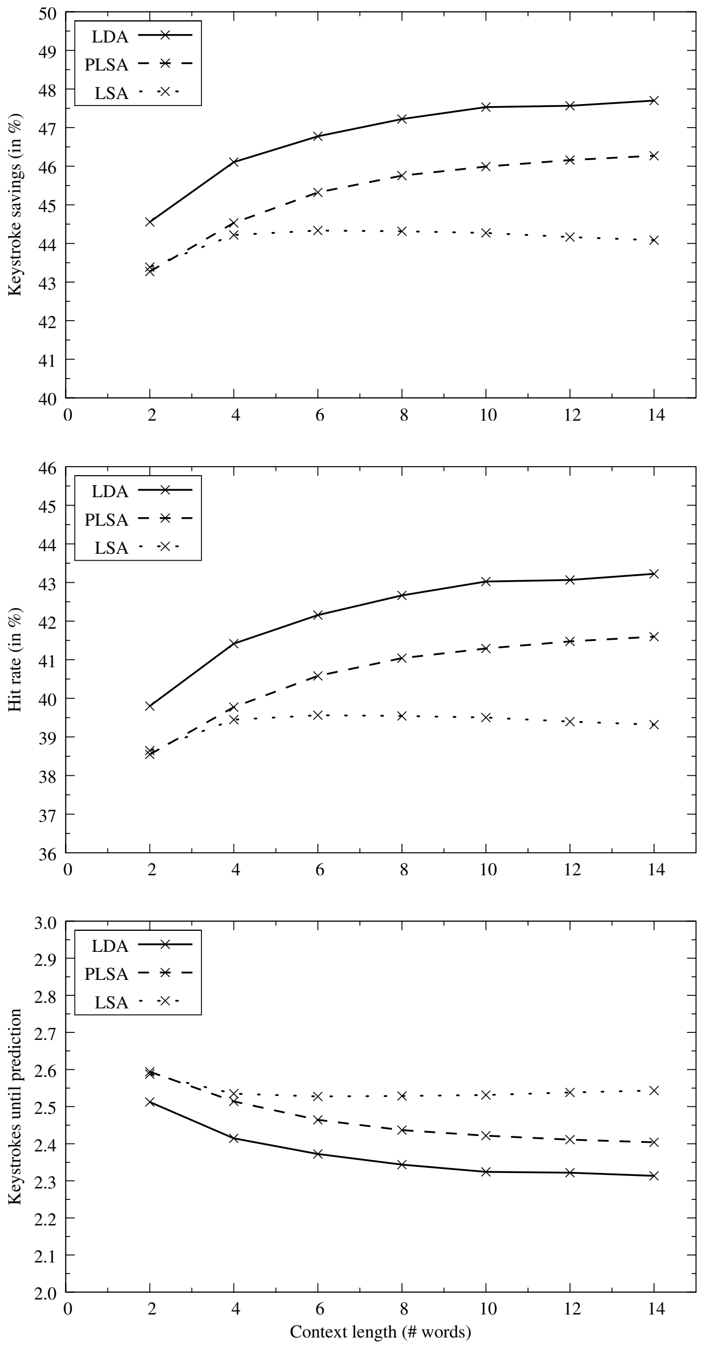

**Fig. 2.** Simulation results for $K = 25$ topics with keystroke savings (top), hit rate (middle), and keystrokes until prediction (bottom) as a function of context length

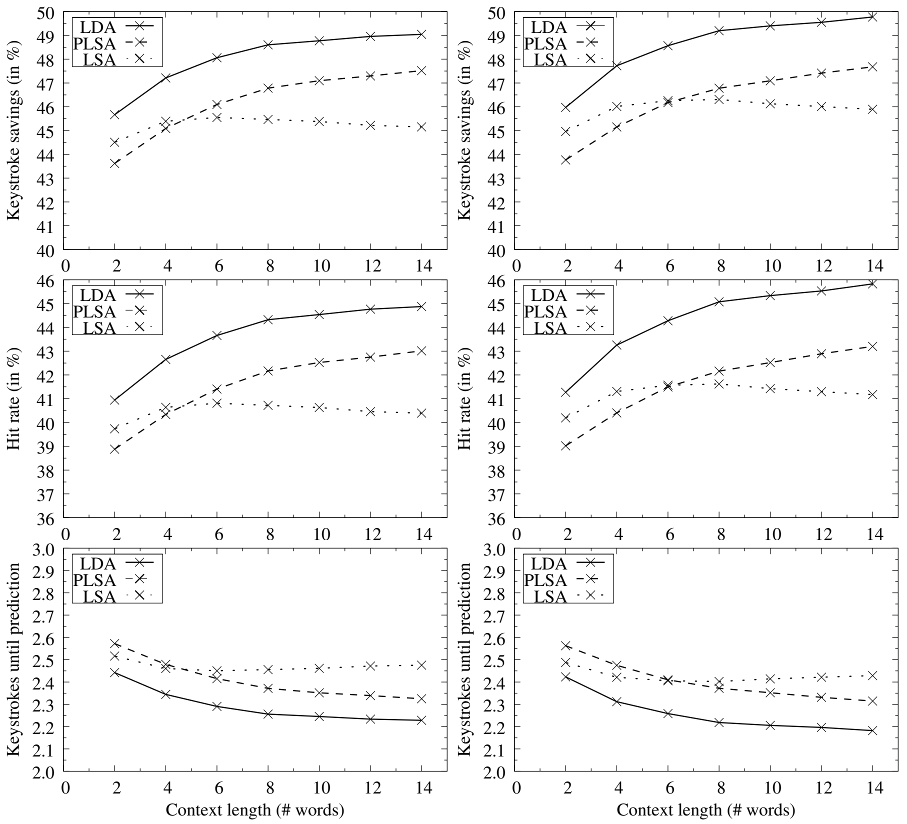

**Fig. 3.** Simulation results for $K = 50$ (left) and $K = 75$ (right) topics with keystroke savings (top), hit rate (middle), and keystrokes until prediction (bottom) as a function of context length

## 6 Conclusions

We have demonstrated the application of probabilistic models like PLSA and LDA, also called topic models, to semantic-based word completion. It has been proved elsewhere that these models are superior to their classical counterpart LSA for semantic modeling of text documents, and our experimental results corroborate their use for applications like word completion. In all our experiments, we found that LDA performed better in predicting or completing words when compared to PLSA and LSA. We also observed that there is a possibility for PLSA to overfit with increasing number of topics and that having too many words in the contextual information might not yield improved results.

We would like to point out again that we restricted ourselves to discrete inputs by simply using word counts in the term–document matrix. It would be interesting to consider extensions of these models for continuous or other non-multinomial data. This would make the models amenable to the standard tf–idf representation of text documents. Our word prediction approach suggests words based on the exclusive use of semantic knowledge. An extension to this approach would be to also integrate syntactic and statistical information to improve the efficiency of the system. Another possible extension is to make these models handle dynamic updates. This is necessary if a typed word is not part of the term–document matrix, thereby making it unpredictable. The possibility that a certain number of folding-in processes might degrade the latent semantic structure should be considered. Dynamic model updates is a non-trivial operation, though there exist some efficient SVD algorithms [17] that could be used for our LSA approach.

## Acknowledgements

We thank Florent Monay (IDIAP) and David Blei (Princeton University) for making their code on PLSA and LDA available online.

## References

1. Swiffin, A., Arnott, J., Pickering, J., Newell, A.: Adaptive and predictive techniques in a communication prosthesis. AAC: Augmentative and Alternative Communication **3** (1987) 181–191
2. Newell, A.F.: Effect of the PAL word prediction system on the quality and quantity of text generation. AAC: Augmentative and Alternative Communication **8** (1992) 304–311
3. Fazly, A., Hirst, G.: Testing the efficacy of part-of-speech information in word completion. In: Proceedings of the Workshop on Language Modeling for Text Entry Methods at the 10th Conference of the European Chapter of the Association for Computational Linguistics, Budapest, Hungary (2003)
4. Kozima, H., Ito, A.: A scene-based model of word prediction. In: Proceedings of the International Conference on New Methods in Language Processing (NeMLaP), Ankara, Turkey (1996) 110–120
5. Li, J., Hirst, G.: Semantic knowledge in word completion. In: Proceedings of the 7th International ACM SIGACCESS Conference on Computers and Accessibility. (2005)
6. Miller, G.A.: Wordnet: An on-line lexical database. International Journal of Lexicography **3** (1990) 235–244
7. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society of Information Science **41** (1990) 391–407
8. Wolf, E.: A semantic-based word completion utility using latent semantic analysis. Diplom-Informatik thesis, Department of Technical Sciences, University of Applied Sciences, Oldenburg/Ostfriesland/Wilhelmshaven, Emden (2005)
9. Hofmann, T.: Unsupervised learning by probabilistic latent semantic analysis. Machine Learning **42** (2001) 177–196
10. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet allocation. Journal of Machine Learning Research **3** (2003) 993–1022

11. Blei, D., Jordan, M.: Modeling annotated data. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Toronto, Canada (2003)
12. Sivic, J., Russell, B., Efros, A., Zisserman, A., Freeman, W.: Discovering object categories in image collections. In: Proceedings of the 10th IEEE International Conference on Computer Vision, Beijing, China (2005)
13. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B **34** (1977) 1–38
14. Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., Saul, L.K.: An introduction to variational methods for graphical methods. Machine Learning **37** (1999) 183–233
15. Brand, M.: Incremental singular value decomposition of uncertain data. In: Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark (2002)
16. Lewis, D.D.: Reuters-21578 Text Categorization Test Collection Distribution 1.0 README File v1.3. (2004)
17. Brand, M.: Fast online SVD revisions for lightweight recommender systems. In: Proceedings of the SIAM International Conference on Data Mining, San Francisco, CA, USA (2003)