Semantic Feature Extraction: Reader-Specific Text Document Classification

Heikki Hyötyniemi
Helsinki University of Technology
Control Engineering Laboratory
Otakaari 5 A, FIN-02150 Espoo, Finland
Phone: +358-9-4513327
E-mail: heikki.hyotyniemi@hut.fi

July 27, 1998

Abstract

An approach to automatic modeling of text documents is presented, where 'semantic features' based on contextual dependencies are extracted from the textual data. The model structure has two levels; first, *context categories* are constructed using sentences in the documents as elementary contextual units, and, second, *document categories* are constructed using the lower-level document analysis results as input data. Models on both of these levels are based on a feature extraction scheme, where the features can be interpreted as coordinate axes in the linear high-dimensional space. The models are adaptive, being updated according to what kind of documents have been read, so that the user-specific 'profile' helps to find relevant documents that match the user's personal model.

An implementation of this approach is presented, technical details are discussed, and some results when using the program are reviewed.

Keywords. Data mining, document modeling, web agents, natural language processing, self-organization

1 Introduction

The search machines (AltaVista, etc.) that are based on traditional methodologies are surprisingly efficient tools. These keyword search engines augmented with sophisticated special properties seem to work nicely — sometimes the hit rate is too high or too low, but, let us face it: in compact information retrieval tasks these traditional approaches probably cannot be outperformed. Where is the market for more heuristic, less consistent data mining approaches?

The operation of the concurrent data mining tools is not reliable; at best they give 'innovative' responses, giving new ideas and points of view. When a user actively searches for information, however, reliable operation of the tool is essential: disappointments soon kill the user's interest. In practice, the user prefers consistency, innovativity comes with lower priority. At least at the moment when the new tools are clearly inferior as compared to the traditional tools in everyday tasks, 'low profile' is a surviving strategy: the basic functionality of an information retrieval tool must not be compromized, and the new algorithms can hopefully sometimes give some added value. The 'add-ons' should work transparently aside, not burdening the user; this means that the operation of the algorithms should be unsupervised and autonomous.

This paper presents a new approach to document modeling: 'semantic features' are extracted from a set of textual documents, and these features are mapped in a self-organizing manner. The automatically generated document models can be utilized to bring added value in traditional tools like news readers.

2 Capturing semantics

Text documents are (or they should be) collections of fresh and meaningful ideas. It is intuitively clear that if one is to classify this kind of documents, the classification must be based on the contents, not on any formal syntactic properties. It is *semantical* analysis that is necessary.

Speaking of 'semantics' here is, of course, not strictly justified; only a human can claim to understand all the connotations that are related to different concepts (and still a person's world view is subjective). When automating text document analysis, it is clear that the standards must be set on a lower level — but even if the problem is difficult, one should not give up immediately.

What we are now studying is *computational* or *associative* semantics — having a very restricted view of the 'meaning'. It is assumed that

> *context directly defines semantics.*

In concrete terms, speaking of words in a natural language, the environment where the words are usually encountered defines their meaning in a pragmatic sense. This engineering-like approach makes it possible to carry out automatic processing of text documents at some limited level of intelligence. The semantics has now a 'floating grounding' — there is no connection to the real world, the words themselves are the lowest-level constructs that are input in the system[1].

In the document modeling approach that was presented in [4], the context was assumed to be captured in successive *three letters* (much too little, clearly!) and in WEBSOM (for example, see [2] and [3]), the context is *three words.* Now, however, it is assumed that a whole *sentence* is the basic semantically relevant contextual unit, conveying exactly one semantically meaningful utterance and defining an 'atomic context'.

3 Linguistic aspects

The above assumption of sentence as the contextual elementary entity is linguistically rather plausible, but the problem that emerges is that a normal, longish sentence has various structurally more or less separate subparts each of them spanning distinct 'flavours' in the 'semantic space'. This structured nature of the contextual units must be taken into account — the meaning of a sentence cannot be efficiently modeled if it is regarded as a massive block with no fine structure, that means, if only the contextual *prototype* is modeled.

This fine structure can be captured using a *feature model,* where the semantical dimensions are represented by feature vectors. It turns out that the GGHA algorithm [6] can be utilized to find the independent semantic components. What are these 'independent components' in this case? If there are documents in very different domains or in different styles (or in different languages) there will be features representing the category centers; other features modify these prototypes in different ways, hopefully in a semantically meaningful way. There can emerge some trivial features, also: if the semantically irrelevant words like 'the', etc., are not eliminated, separate features are needed to track them.

Due to pragmatical reasons, no morphological or grammatical preprocessing is applied to the sentences before they are input in the algorithm: sentences are seen simply as unordered sets of words. It is questionable whether the 'flattening' of the relations, so that all dependencies become reciprocal, is really justified[2]. The contents of the word categories is dictated only by *relevance,* the co-occurrence of words, and, hopefully, after adaptation the categories carry functionally independent roles. This view of the categories resembles the ideas that are adopted in *functional linguistics.*

In [7], trends in natural language understanding are reviewed. One of the most influential paradigms is that of *transformational grammars* and the 'deep structures' underlying the surface form of utterances [1]. These deep structures are assumed to be strictly syntactic, so that semantic tags can be added in the structures afterwards. This view of separated syntactic and semantic analyses has been criticized, but syntax still has a major role in all linguistic theories.

[1] To control the convergence of the contextual categories (as will be discussed later), there would be a rather 'brute force' method: the *sets of synonyms* could be learned together, so that the words with approximately the same meaning would finally be found in the same categories, and these categories could be labeled accordingly. However, the problem of *infinite recess* cannot be avoided when applying only symbolic concepts

[2] In this idyllic world the sentence 'boy loves girl' is equivalent to the sentence 'girl loves boy' ...

The approach that is adopted here assumes *no syntax, only semantics!* It must be remembered that this starting point has only pragmatic motivation. However, it would be interesting to study the psycholinguistic consequences of ‘deeper structures’, the above presented semantic features as the fundamental elements of cognition ... One problem that is immediately evident is that because of the missing relational structure between the constructs, it would be difficult to define one-to-one transformations between the inner constructs and the surface structure.

4 Analysis of documents

A complete text document is such a complex entity that (at least) *two analysis levels* are needed. In this approach, a *context category model* is first constructed, containing the extracted contextual sentence-level features that were discussed above. Based on this lower level sentence-wise analysis, a ‘fingerprint’ is constructed for the whole document: this fingerprint contains the *category distribution* for the document, revealing how frequently each of the categories is encountered in the document. The second-level categorization, or the *document category model,* is based on these document fingerprints.

Now, there is a catch again: the documents are at least as many-faceted as the elementary sentences are, and there are usually hints towards many different contextual dimensions. That is why, the feature extraction scheme is applied also in this level (see Fig. 1). Comparing to other applications of the presented feature extraction scheme (see [5]), this application field is more complex — and it is also the first example where *multi-level analysis* is needed.

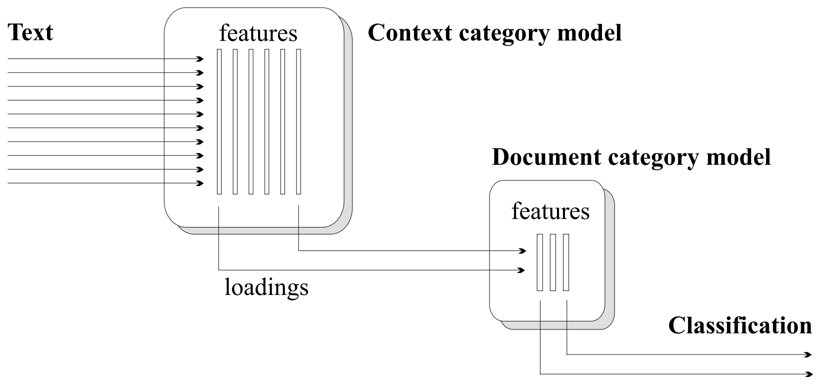

Figure 1: Schematic diagram of the two-level text document analysis (note that there is context category fitting for each sentence, whereas the document category fitting is done only once for one document

In the experiments, the (adaptive) set of 10000 most commonly used words (or letter combinations) is first compressed to 100 to 256 context categories, based on the sentence-level information, and after that, 8 to 16 document categories are constructed based on the document-level information. The problem of combining temporally structured information (the ‘one-dimensional’ flow of words in the document) has now been solved in a trivial way, neglecting the temporal structure: first, the words within a sentence are processed as a non-ordered list, and, second, the fingerprint of the whole document is constructed by simply adding the elementary sentence categorization results together, with no emphasis on their order of appearance.

The articles that are read are used for adapting the categories. That is why, after a while the document model becomes a reader-oriented ‘profile’ because the categories are adapted to optimally match the reader’s taste. This means that the reader obtains a personal ‘profile’ for the documents in his own ‘interest space’.

5 Some results

The presented modeling approach has been applied to implement an article reader for the Internet news groups (compare to [2]). Only very preliminary test results are available this far. However, these results seem rather promising.

It was two news groups that were modeled, `comp.ai.nat-lang` about natural language processing and `comp.ai.philosophy` about general topics on AI, both groups containing several hundred documents. The input dimension of the word category model was 10000, and 5 out of 100 features were used to describe the sentences; in the document category model, the input dimension was (naturally) 100, and 3 of 9 features were used to describe the documents.

When the two models had converged, they were tested by browsing through the same news groups again (even if mixing the training set and the test set together is a major mistake, now it was done; it is the classification capability that is tested, and there was never any critic giving 'correct' classifications). First, the distribution of the document fingerprints (the normalized sum of loadings for all word category features in a document) in both news groups are shown in Figs. 2 and 3 (the white curve representing the mean values). In Figs. 4 and 5, the weight of one of the 9 document features is shown for 550 first documents in both news groups. It seems that at least the groups can be identified pretty well!

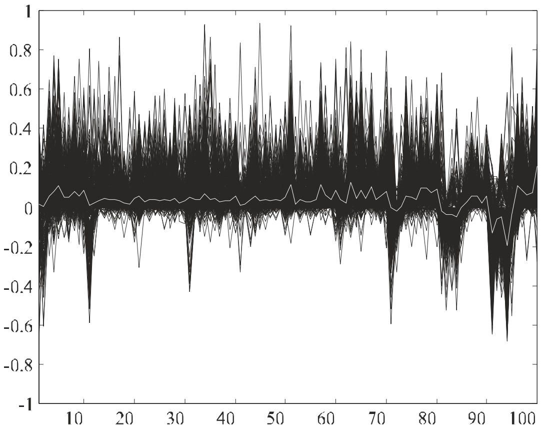

Figure 2: The fingerprints in the news group `comp.ai.nat-lang`

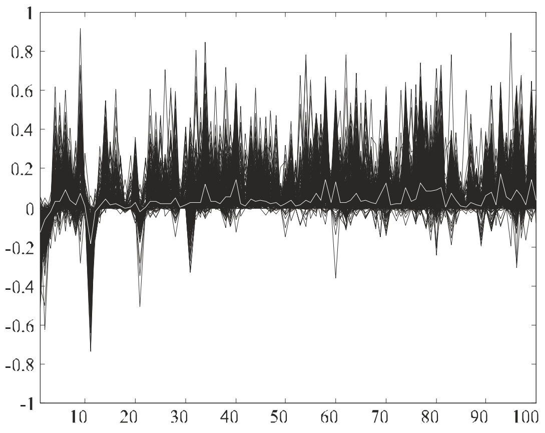

Figure 3: The fingerprints in the news group `comp.ai.philosophy`

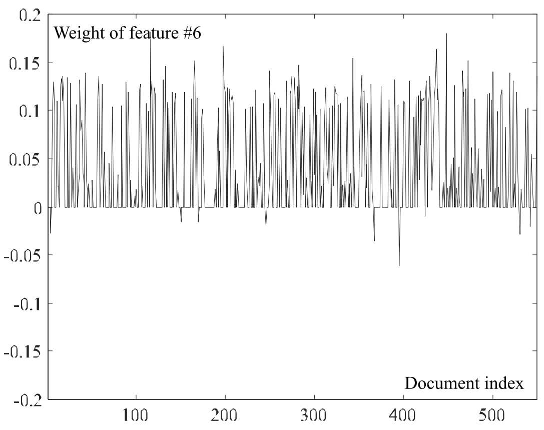

Figure 4: Loading of feature #6 for 550 documents in `comp.ai.nat-lang`

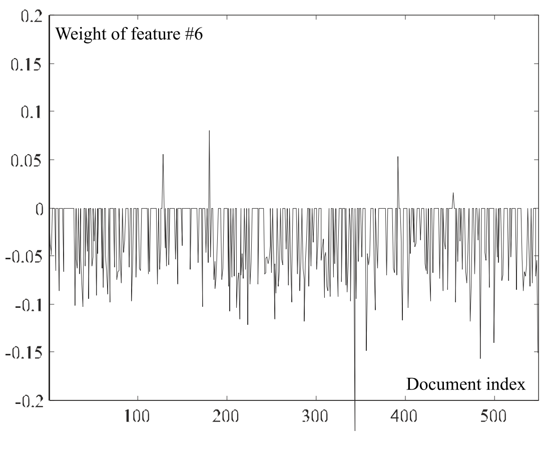

Figure 5: Loading of feature #6 for 550 documents in `comp.ai.philosophy`

6 Using the program

During a session, new documents from the selected news group are fetched, these documents are matched against the models, and various analysis results are printed next to the document header information accordingly: the percentage of strange words in the document, the overall model matching error, and the actual classification in compact form. If the user selects some of the documents (by clicking the mouse), the document is shown to him; after that, the user is asked to press a button if that document is relevant and interesting from his point of view — if it is, the models are adapted using that document as input data to better match the user's interests.

It is easy to think of enhancements to this basic operation of the program; at the moment, the only goal has been to achieve the level of basic functionality.

The parameters controlling the adaptation process are defined in a configuration file. Before starting the session, these values can be modified (for example, if some totally new group of documents is to be modeled, it may be reasonable to adjust the parameters to achieve faster adaptation).

The program will be available at WWW address `http://saato014/Hyotyniemi/publications/98_step` for free experimenting (note that the version number is still 0.99!). The program should be compatible with Microsoft Windows 95/NT systems with at least 16 megabytes of memory, and the harddisk requirement is something like 10 MB. Files containing prototypical context and document category models are also included in the package.

7 Technical details

7.1 About the implementation

The preprocessing of the text material is minimal, only simple character-level manipulations are applied. For example, non-letters are eliminated; punctuation marks that are used for ending sentences are substituted with periods, whereas all other special characters are simply dropped (or substituted with white spaces).

The extracted words are stored in a special tree-form data structure, where each node represents a letter; searching for a word in the tree is rather efficient, and memory requirements are minimized. Words that are stored in the tree have unique index numbers, and later on, only these indices are used when referring to the words: these indices directly reveal the corresponding input vector entries. Because no morphological analysis or any other validity check for the input words is accomplished, the tree needs to be pruned every now and then. When the tree capacity is exhausted, the least frequent words are eliminated (currently this pruning takes place only when the models are loaded into memory).

Even if the data models are rather sparse, that is, there are plenty of zero values in the feature vectors, the data structures are implemented as full matrices for pragmatic reasons. Specially the word category model is extensive: if there are 10000 words and if 100 word categories are to be extracted (these are the practical minimums!), one million 'float' numbers need to be simultaneously kept in memory, meaning that four megabytes are exhausted. Approximately the same amount of harddisk memory is used for storing this model. However, using today's computer facilities, these figures do not yet sound too bad.

The GGHA algorithm that is applied for constructing the context category model and the document category model, is slightly modified: for example, the extracted features are typically *more non-orthogonal* than when using the standard algorithm [6] (how this is accomplished is not elaborated on here). Another additional feature that is added in the document category modeling is the weighting of the input based on the document length; it is evident that short documents may have statistically very peculiar properties, and longer documents should be trusted more when the model is adapted.

No other weightings based on *a priori* probabilities are applied in the algorithms, even if that might be beneficial; for example, the disturbance effect caused by the most frequent (and least specific) words could be compensated using the frequencies for determining the values in the weighting matrix W — see [6] for the algorithm (now the elements in this matrix are binary, either ones or zeros).

7.2 Efficiency aspects

To be used on-line, the efficiency requirements are rather severe. The most critical part of the system is the word category model, because of the high dimensionality of the input vector and because this model is needed after each sentence, that means many times during processing of a single document.

The algorithm is presented in [6], available in this Proceedings, and it will not be rewritten here; however, there are some special aspects concerning the efficiency in this particular application that deserve special attention. When trying to avoid manipulating the 10000–20000 dimensional word category vectors, the key point is the weighting matrix W: now W will only have nonzero elements if the corresponding word exists in the sentence to be processed. This results in 'local' processing — in the adaptation of the vectors, only those rows need to be touched that are explicitly used.

The normalization of the feature vectors still has to be done after each step, and it would seem that here one cannot avoid going through the feature vectors element by element. However, now it is not the actual feature vector θ_k (k being now time index) that is stored, but an unnormalized version of it, or $\tilde{\theta}_k$, so that $\theta_k = \alpha_k \tilde{\theta}_k$, where α_k is a scalar. If one wants to update the feature vector by adding a (sparse) vector $\Delta\theta_k$ to it, using the unnormalized versions this can be accomplished simply as $\tilde{\theta}_{k+1} = \tilde{\theta}_k + \frac{1}{\alpha_k} \cdot \Delta\theta_k$. To update the scaling factor α_k, so that $\|\alpha_{k+1}\tilde{\theta}_{k+1}\| = 1$ if also $\|\alpha_k \tilde{\theta}_k\| = 1$, the following law can be derived:

$$\alpha_{k+1} = \frac{\alpha_k}{\sqrt{1 + 2\alpha_k \tilde{\theta}_k^T \Delta\theta_k + \Delta\theta_k^T \Delta\theta_k}}.$$

If $\Delta\theta_k$ has only a few nonzero elements, these calculations are simple to carry out. It turns out that none of the steps in the algorithm are dependent of the feature vector size, so that the complexity factor is of the order $\mathcal{O}(\log n)$ only; this is caused by the scanning through the word tree.

Acknowledgements

I am grateful to Mr. Dan Lumenko who implemented the preliminary user interface and the Internet connection in the program. This study was financed by the Academy of Finland.

References

[1] Chomsky, N.: *Syntactic Structures.* Mouton, The Hague, 1957.

[2] Honkela, T., Kaski, S., Lagus, K., and Kohonen, T.: *Newsgroup Exploration with WEBSOM Method and Browsing Interface.* Helsinki University of Technology, Laboratory of Computer and Information Science, Report A32, Espoo, Finland, 1996.

[3] Honkela, T., Kaski, S., Kohonen, T., and Lagus, K.: *Self-Organizing Maps of Very Large Document Collections: Justification for the WEBSOM method.* In "Classification, Data Analysis, and Data Highways" (eds. Balderjahn, I., Mathar, R., and Schader, M.), Springer-Verlag, Berlin, 1998, pp. 245–252.

[4] Hyötyniemi, H.: *Text Document Classification with Self-Organizing Maps.* In the Proceedings of Finnish Artificial Intelligence Conference (STeP'96), Vaasa, Finland, August 20-23, 1996, pp. 64–72. Available at `http://saato014/Hyotyniemi/publications/`.

[5] Hyötyniemi, H.: *On the Statistical Nature of Complex Data.* In "SCAI'97 — Sixth Scandinavian Conference on Artificial Intelligence: Research Announcements" (ed. Grahne, G.), Helsinki University, Department of Computer Science, Report C–1997–49, Helsinki, Finland, 1997, pp. 13–27. Available at `http://saato014/Hyotyniemi/publications/`.

[6] Hyötyniemi, H.: *On Mental Images and 'Computational Semantics'.* In the Proceedings of Finnish Artificial Intelligence Conference (STeP'98), Jyväskylä, Finland, September 7–9, 1998. Available at `http://saato014/Hyotyniemi/publications/`.

[7] Moyne, J.A.: *Understanding Language: Man or Machine.* Plenum Press, New York, 1985.