Learning to Segment and Label Semi-Structured Documents with Little or No Supervision

Sander Canisius S.V.M.Canisius@uvt.nl
Caroline Sporleder C.Sporleder@uvt.nl
ILK / Communication and Information Sciences, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands

Abstract

We present two machine learning approaches to information extraction from semi-structured documents that can be used if no annotated training data are available but there does exist a database filled with information derived from the type of documents to be processed. One approach tries to employ standard supervised learning by artificially constructing labelled training data from the contents of the database. Truly knowledge-free production of these artificial data turned out to result in suboptimal classification performance. However, given a small amount of annotated training data, the artificially constructed data can be modelled to better match the structure of target documents, causing considerable performance improvement. The second approach combines unsupervised Hidden Markov Modelling with language models. Training such a system requires nothing but the information available in the database. Empirical evaluation of both systems pointed out that the Hidden Markov Model managed best to learn the task of segmenting and labelling biological field book entries from a derived database only. Further analysis showed that its performance could be improved even more if the initial segmentation of field book entries is optimised.

1. Introduction

Over the past decades much textual data has become available in electronic form. Many text types, such as lists of classified advertisements, medical records, or logs of zoological expeditions, adhere to some inherent structure. Typically, such documents are made up of a number of shorter texts (or *entries*), each describing an individual object (e.g., an appartment, a zoological find) or event (e.g., a patient presenting to a health care provider). These descriptions in turn typically consist of different segments (or *fields*) which contain information of a specific type drawn from a more or less fixed inventory. Example (1), for instance, shows two descriptions of zoological specimens (a snake and three frogs) collected during an expedition. The descriptions consist of fields giving information about the specimens and the circumstances of their collection. For example, in the first description, *Leptophis* and *ahaetulla* refer, respectively, to the genus and species of the specimen, *road to Overtoom* mentions the place of collection, *in bush above water* encodes information about the biotope, *in the process of eating Hyla minuta* is a remark about the circumstances of collection, *16-V-1968* gives the collection date and *RMNH 15100* the registration number.

(1) Leptophis ahaetulla, road to Overtoom, in bush above water in the process of eating Hyla minuta 16-V-1968. RMNH 15100

Hyla minuta 1 ♀ 2 ♂ Las Claritas, 9-VI-1978 quaking near water 50 cm above water surface, near secondary vegetation, 200 m, M.S. Hoogmoed, RMNH 27217 27219

Unfortunately, this inherent structure is rarely made explicit. While the different object or event descriptions might be indicated by additional whitespace or formatting means, as in the example above, the individual fields within a description are typically not marked in any way. However, knowledge of the field structure would be very beneficial for information extraction and retrieval. For instance, texts in their raw form only allow key word search. To retrieve all entries describing specimens of type *Hyla minuta* from a zoo-

Proceedings of the 18th Benelearn
P. Adriaans, M. van Someren, S. Katrenko (eds.)

logical field report, one can only search for occurrences of that string anywhere in the document. This can return false positives, such as the first description in (1) above, which does contain the string but is not about a *Hyla minuta* specimen but about a specimen of type *Leptophis ahaetulla* (the string *Hyla minuta* just happens to occur in the SPECIAL REMARKS field). On the other hand, if the genus and species information in an entry was explicitly marked, it would be possible to query specifically for entries whose GENUS is *Hyla* and whose SPECIES is *minuta*, thus avoiding the retrieval of entries in which this string occurs in another field.

The task of automatically finding and labelling segments in semi-structured texts has been referred to as *field segmentation* (Grenager et al., 2005). It differs from information extraction tasks of the "template-filling" variety in that all or most of the information in the input document is assumed to be relevant and the goal is to segment the document into fields containing different types of information. Field segmentation can be modelled as a sequence labelling problem, where each text is viewed as a sequence of tokens and the aim is to assign each token a label indicating to which field the token belongs (e.g., BIOTOPE). If data in the form of texts annotated with segment information was readily available, the problem could be approached by training a sequence labeller in a supervised machine learning step. However, manually annotated data is often not readily available. Creating it from scratch is time consuming and often requires a certain amount of expert knowledge. Moreover, the sequence labeller has to be re-trained for each new domain (e.g., natural history vs. archaeology) and possibly also each sub-domain (e.g., insects vs. mammals, different periods in history) due to the fact that the relevant fields vary from domain to domain. For example, a report describing a collection of fish may contain information about the distance between the place where a specimen was caught and the nearest coast, which is obviously irrelevant for other types of animals.

Thus, fully supervised machine learning is not feasible for this task. In this paper, we explore two approaches which require no or only a very small amount of manually labelled training data. Both approaches exploit the fact that there are often structured resources *derived* from the original, semi-structured documents that can potentially be utilised to bootstrap a sequence labeller. It is common practice, for example, that information contained in field reports or medical records is manually entered into a database, usually in an attempt to make the data easier to search. In such databases, each row corresponds to an entry in the original document (e.g., a zoological specimen), and the database columns correspond to the fields one would like to discern in the original document. Manually converting raw text documents into databases is a laborious task though, and it is rather common that the database covers only a small fraction of the objects described in the original texts. This is certainly the case in the cultural heritage domain, on which we focus in this paper, where database extension is not a high-priority task for which extra staff is employed but is done by researchers and collection curators whenever their other duties permit it. The research question we address in this paper is whether it is possible to bootstrap a domain-specific field segmentation system from a small, manually created database for that domain. Such a system could then by applied to the remaining texts in that domain, which could then be segmented (semi-)automatically and possibly be added to the original database. This approach is easily portable to new domains, provided that there are existing databases for that domain.

A database does not make perfect training material for a field segmenter though, as it is only derived from the original document, and there are typically significant (and sometimes systematic) differences between the two data sources: First, while the ordering of the segments in a semi-structured text document is often not entirely fixed, some orderings are more likely than others. This information is lost in the derived databases. Second, the databases may contain information that is not normally present in the underlying text documents, for example information relating to the storage of an object in a collection. Conversely, some of the details present in the texts might be omitted from the database, e.g., the SPECIAL REMARKS field might be significantly shortened. Third, pieces of information are frequently re-written when entered in the database, in some cases these differences may be systematic, e.g., dates, person names, or registration numbers might be written in a different format. Despite of this, these databases will provide certain clues about the structure and content of different fields. We exploit this in two different ways: (i) by concatenating database fields to artificially create annotated training data for a supervised machine learner, and (ii) by using the database to build language models for the field segmentation task.

2. Related Work

Most approaches to field segmentation and related information extraction tasks, such as template filling, have been supervised. Freitag and Kushmerick (2000) combine a pattern learner with boosting to perform

field segmentation in raw texts and in highly structured texts such as web pages and test this approach on a variety of field segmentation and template filling tasks. Kushmerick et al. (2001) address the problem of extracting contact information from business cards. They mainly focus on field *labelling*, bypassing the segmentation step by assuming that each line on a business card only contains one field (though a field like ADDRESS may span several lines). Their method combines a text classifier, for assigning likely labels to each field, with a trained Hidden Markov Model (HMM) for learning ordering constraints between fields. Borkar et al. (2001) identify fields in international postal addresses and bibliographic records by nesting HMMs: an outer HMM for modelling field transitions and a number of inner HMMs for modelling token transitions within fields. Viola and Narasimhand (2005) also deal with address segmentation but employ a trained context-free grammar.

An unsupervised approach is provided by Grenager et al. (2005), who perform field segmentation on bibliographic records and classified advertisements, using EM to fit an HMM to the data. They show that an unconstrained model does not learn the field structure very well and propose augmenting it with a limited amount of domain-unspecific background knowledge, e.g., by modifying the transition model to bias it towards recognising larger-scale patterns.

3. Learning Field Segmentation from Databases

3.1. Data

We used two data sources in the experiments: a database containing information about a small part of the reptile and amphibian collection of Naturalis, the Dutch National Museum of Natural History, and several field book descriptions of such specimens. The database was created from the same field books but we made sure that the database entries did not overlap with the field book entries we used in the experiments, i.e., both data sets describe different specimens.

The database consisted of 16,670 entries and 41 columns. Many database cells are empty. Those that are filled come in a variety of formats, i.e., numbers (REGISTRATION NUMBER), dates (COLLECTION DATE), individual words (GENUS), and free text (BIOTOPE). 22 of the columns contained information that was missing from the field books, e.g., information relating to the storage of the specimens; these columns were excluded from the experiments.

We randomly selected 300 field book entries which were not covered by the database and annotated these with segment information. Annotating this amount of data took one person three days. To test the reliability of the manual annotation, 50 entries were labelled by two annotators. The inter-annotator accuracy on the token level was 92.84% and the kappa .92. The number of distinct field types found in the entries was 19, some of which only occurred in two entries, others occurred in virtually every entry. The average field length was four tokens, with a maximum average of 21 for the SPECIAL REMARKS field, and a minimum of one for fields such as SPECIES or GENUS. The average number of tokens per entry was 60. Punctuation tokens that did not clearly belong to any field were labelled as OTHER. For the experiments, we split the annotated entries into a development set of 100, which we used for parameter tuning, and a test set of 200.

3.2. Baselines

In order to get a sense of the difficulty of the task at hand, we implemented five baseline approaches. For the first, **Majority** (**MajB**), we always assign the field label that occurs most frequently in the manually labelled test data, namely SPECIAL REMARKS. The other four baselines implement different look-up strategies, using the database to determine which label should be assigned to a token or token sequence.

Exact (**ExactB**) looks for substrings which exactly match the content of a database cell and then assigns each token in the matched string the corresponding column label from the database. There are normally several ways to match a field book entry to the database cells; we employed a greedy search, labelling the longest matching substrings first. All tokens that could not be matched were assigned the label OTHER.

Unigram (**UniB**) assigns each token the column label of the database cell in which it occurs most frequently. If a token is not found in the database, it is labelled as OTHER.

Trigram (**TriB**) assigns each token the most frequent column label of the trigram centred on it. If a trigram is not found in the database, the baseline backs off to the two bigrams covering the token and then to the unigram. If the token is not found in the database, OTHER is assigned.

Trigram+Voting (**TriB+Vote**) is based on a technique proposed by Van den Bosch and Daelemans (2005) for sequence labelling tasks. The main idea is to assign labels to trigrams in the sequence using a sliding window. Because each token, except the boundary tokens, is contained in three different trigrams (i.e.,

the one centred on the token to its left, the one centred on itself, and the one centred on the token to its right), each token gets three labels assigned to it, over which voting can be performed. In our case the labels are assigned by database look-up. If a trigram is not found in the database, no label is assigned to it. If the labels assigned to a given token differ, majority voting is used to resolve the conflict. If this does not break the tie (i.e., because all three trigrams assign different labels), the label of the trigram that occurs most frequently in the database is assigned. We also implemented two post-processing rules: (i) turning the label OTHER between two identical neighbouring labels into the surrounding labels, and (ii) labelling commas as OTHER if the neighbouring labels are not identical.

3.3. Supervised Learning from Automatically Generated Training Data

Our first strategy was to automatically generate training data for a supervised machine learner from the database. Since the rows in the database correspond to field book entries and the columns corresponds to the fields that we want to identify, training data can be obtained by concatenating the cells in each database row. The order of the fields in the field book entries is not fixed and this should also be reflected in the artificially generated training data. However, the field sequence is not entirely random, i.e., not all sequences are equally likely. If a small amount of manually annotated data is available, the field transition probabilities can be estimated from this, otherwise the best one can do is to assume uniform probabilities for all possible orderings. We experimented with both strategies, creating two different training sets, one in which the database cells were concatenated randomly with uniform probabilities, and another in which the cells were concatenated to reflect the field ordering probabilities estimated from ten entries in the manually labelled development set.[1] When estimating the field transition probabilities, we computed a probability distribution over the initial fields of an entry as well as the conditional probability distributions of a field x following a field y for all seen segment pairs in the ten entries. To account for unobserved events, we used Laplace smoothing.

[1]We found that 10 annotated entries are enough for this purpose; the results we obtained by estimating the sequence probabilities from 100 entries were not significantly different. This is probably because the probabilities are only used indirectly to bias the field orderings for the generated training data. If the probabilities were used directly in the model, the amount of manually annotated data would probably matter much more.

The artificially created training data were then converted to a token-based representation in which each token corresponds to an instance to be assigned a field label. On the whole, we had just under 700,000 instances (i.e., tokens) in our training data. We implemented 107 features, falling in three classes:

- the neighbouring tokens (in a window of 5 centering on the token in focus)
- the typographic properties of the token (word vs. number, capitalisation, number of characters etc.)
- the tfidf weight of the token in its context with respect to each of the columns in the database

The tfidf based features were computed for a window of three, centering on the token in focus. For all n-grams in this window covering the token in focus (i.e., the trigram, the two bigrams, and the unigram of the focus token), we calculated the $tfidf$ similarity with the columns in the database, where the similarity between an n-gram t_i and a column col_x is defined as:

$$tfidf_{t_i,col_x} = tf_{t_i,col_x} \ \log \ idf_{t_i}$$

The term frequency, tf_{t_i,col_x} is the number of occurrences of t_i in col_x divided by the number of occurrences of all n-grams of length n in col_x (0 if the n-gram does not occur in the column). The inverse document frequency, idf_{t_i}, is the number of all columns in the database divided by the number of columns containing t_i. A high tfidf weight for a given n-gram in a given column means that it frequently occurs in that column but rarely in other columns, thus it is a good indicator for that column.

The data was then used to train a memory-based machine learner (TiMBL (Daelemans et al., 2004), default settings, $k = 3$, numeric features declared) to determine which field each token belongs to.

3.4. Hidden Markov Models

Our second approach combines language modelling and Hidden Markov Models (HMMs) (Rabiner, 1989). Hidden Markov Models have been in use for information extraction tasks for a long time. A probabilistic model is trained to assign a label, or *state* to each of a sequence of observations, where both labels and observations are expected to be sequentially correlated; hence the popularity of HMMs in natural language processing and information extraction. Recently, a large number of more sophisticated learning techniques have largely replaced HMMs for information extraction; however unlike most of those newer techniques, HMMs offer the advantage of having a well-established

unsupervised training procedure: the Baum-Welch algorithm (Baum et al., 1970).

Training a Hidden Markov Model, whether supervised or unsupervised, comes down to estimating three probability distributions.

1. An initial state distribution π, which models the probability of the first observation of a sequence to have a certain label.
2. A state-transition distribution A, modelling the conditional probability of being in a certain state s, given that the previous state was s'.
3. A state-emission distribution B, which models the conditional probability of observing a certain object o given some state s.

For information extraction tasks, the typical interpretation of an *observation* as referred to above, is that of a token, where the entire observation sequence commonly corresponds to one sentence. In the current study, we chose to apply HMMs on a somewhat higher level, where an observation corresponds to a *segment* of the field book entry. Ideally, one such segment maps one-to-one to a cell in the specimen database, though we leave open the possibility of merging several segments into one database cell.

Provided that a field book entry can be segmented reliably, we have turned one part of the learning problem, that of estimating the state-emission distribution, into one for which we have (almost) perfect supervised training data: the contents of the database cells. The general form of a Hidden Markov Model's state-emission distribution is $P(o|s)$, where s is the state, i.e. a field type in our case, and o is the observation. As mentioned before, we treat a segment of tokens as one observation, therefore our state-emission distribution will look like $P(o = t_1, t_2, ..., t_n|s)$. Essentially, what we have here is a language model, conditioned on the current state. Since the specimen database provides a large amount of labelled segment sequences, any probabilistic language modelling method can be used to estimate the state-emission distribution.

Whereas the specimen database provides sufficient information to estimate the state-emission distribution in a fully supervised way, the initial-state and state-transition distributions cannot be derived from the database alone. Columns in a database are either unordered or ordered in a way that does not necessarily reflect the order they had in the field book entries they were extracted from. However, the original field book entries do show a rather systematic structure. Often, using information about the order fields typically occur in, seems to be the only way to distinguish certain field types from one another. To estimate the two missing probability distributions, the Baum-Welch algorithm was used, updating the initial-state and state-transition distributions, while keeping the state-emission distributions unchanged.

3.4.1. Segmentation of Field Book Entries

In our setup, the Hidden Markov Model expects the input texts to be presegmented. To come up with a good initial segmentation of an input entry, we again chose a language-modelling approach. It is expected that segment boundaries can best be recognised by looking for unusual token subsequences; that is, token sequences that are highly unlikely to occur within a field according to the information we obtained from the specimen database about what a typical segment does look like. An n-gram language model has been trained on the contents of *all* the columns of the specimen database. Using this language model and the Viterbi algorithm, the globally most-likely segmentation of the input text is predicted.

3.4.2. The State-emission Model

The state-emission model is constructed by training a separate language model for each column of the specimen database. Combining those gives us the conditional distribution required for a Hidden Markov Model. However, in the specimen database, not every column has actually been filled for every record. There are columns that only contain actual data as infrequently as in 5% of the records. Relative to columns that contain data more often, these sparsely-filled columns tend to be overestimated when simply computing a likelihood according to the language model. For this reason, a penalty term is added to the state-emission distribution corresponding to the probability that a record contains data for the given column. The likelihood computed by the language model and the corresponding penalty term are then simply multiplied.

3.4.3. Language Modelling

For building both types of language model presented in the two previous sections, we used n-gram language modelling as implemented by the SRI Language Modelling Toolkit (Stolcke, 2002). With this toolkit, high-order n-gram models can be built, where the sparsity problem often encountered with such models is tackled by various smoothing methods. We supplemented this built-in n-gram smoothing, with our own smoothing on the token level by replacing low-frequent words

with symbols reflecting certain orthographic features of the original word, and numbers with a symbol only encoding the number of digits in the original number.

In addition to these general measures to deal with sparsity, we also applied a small number of knowledge-driven modifications to the training data for the language models. The need for those is caused by the fact that the contents of the specimen database are almost, but not entirely extracted literally from the original field book entries. For example, in the field books, mentions of dates have often been transcribed with Roman numerals for month numbers; the specimen database, however, encodes all months using regular Arabic numerals. As a consequence of this, a date model trained naively on the contents of the database, would most likely fail to recognise many dates in the field book entries. For this specific case, we randomly changed some of the month numbers in the training data to Roman numerals.

Another difference between the field books and the database that turned out to be rather crucial is the fact that many segments in the field book entries are separated by commas. Such commas used as delimiters have not been copied to the database. However, commas do occur in the database, since in many field types —especially the SPECIAL REMARKS field— commas are used for purposes other than marking the end of the segment and the start of a new one. To deal with this difference, we modified the training data for the segment model by randomly inserting commas at the end of segments. Experimental results point out that this modification has a large impact on the performance of the segmentation model.

3.5. Results and Discussion

To evaluate the performance of the two approaches, we set up a series of experiments in which the contents of the database were the only training data, and 200 annotated field book entries served as test data. As the only exception to this, 10 annotated field book entries, not overlapping with those used for testing, were used for estimating the field ordering probabilities for generating the supervised training data for the memory-based learner. For the memory-based learning approach, training meant creating artificial data from the database cells and subsequently training a memory-based learner on these data.

For the Hidden Markov Model, training consisted of (1) estimating a language model to serve as a segmentation model, (2) estimating separate language models for each database cell to estimate state-emission probabilities, and (3) performing Baum-Welch optimisation on the (unlabelled) test data. Being an unsupervised training method, Baum-Welch can easily be applied to the test data, even though it does not have access to label information for them. We hypothesised that applying it to the test data themselves rather than to an arbitrary unlabelled data set would result in a better model for the data at hand. This is in fact rather similar to transductive learning, which also produces models specifically optimised for performing well on a given test set.

Performance of the systems was measured using a number of different metrics, each reflecting different qualities of a segmentation. The most basic one, token accuracy, simply measures the percentage of tokens that were assigned the correct field type. It has the disadvantage that the accuracy does not reflect the quality of the segments that were found. For a more segment-oriented evaluation, we used precision, recall and F-score on correctly identified and labelled segments. For a segment to be counted as correct the boundaries had to be exactly the same as in the annotation of the test data; no credit was assigned for partially identified segments. As a last measure for segmentation quality we used WindowDiff (Pevzner & Hearst, 2002), which only evaluates segment boundaries not the labels assigned to them. In comparison with F-score, it is more forgiving with respect to an occasional incorrectly inserted or omitted segment boundary.

The top half of Table 1 shows the performance of each of the baseline methods described in Section 3.2. While their performance may not be suited for any serious application, they do prove that there is sufficient overlap between the contents of the database and the field book entries to actually learn how to label tokens. This is illustrated even better by the learning curves depicted in Figure 1. The curves start at the point where only 10% of the database records are used for training. This percentage is gradually increased up to the point where the complete database consisting of 16,670 records is used. Clearly, all baseline approaches benefit from having more training data, though the increase in performance with respect to the increase of data is only modest. The biggest problem of all baseline approaches is that their performance with respect to the segment-oriented measures is disappointing. Even the best baseline method, the trigram lookup with voting, only reaches an F-score of 19.4.

Looking at the performance of the two memory-based learners in Table 1 (*MBL rand.* was trained on randomly concatenated training data, *MBL bias* on data

modelled after 10 training sequences), we see that the small amount of prior knowledge used for generating the artificial training data results in a substantial improvement compared with the memory-based learner that was trained on randomly concatenated training data with uniform probabilities. Compared to the best baseline method, there is an improvement in terms of token accuracy. Surprisingly however, the F-score of the memory-based learner (17.6) is worse than the best baseline method's F-score (19.4). This is solely caused by a lower precision of the memory-based learner. Another remarkable observation is the fact that the amount of training data hardly has any influence on the performance of the memory-based learner. As shown by Figure 2, whether only 10% of the database or the complete database is used for constructing training data, the resulting performance is almost the same.

As can be seen in the second-to-last row of Table 1, the Hidden Markov Model outperforms all other approaches in all aspects; it attains both the best token accuracy (60.0), and by far the best F-score (45.4). Looking at the learning curve shown in Figure 2, a small positive effect of increasing the amount of training data can be observed, though the curve is not as smooth as is the case with a typical supervised learner.

Being composed of two interdependent processing stages, the HMM approach may suffer from error propagation, where errors committed in the segmentation stage may cause further errors in the labelling stage. To analyse the effect of this error propagation, we also evaluated the labelling step separately, applying it to perfectly segmented input data. The results of this experiment can be found in the last row of Table 1. In terms of token accuracy, perfectly segment input does not even improve performance much; compared to the HMM scores on predicted segments, token accuracy increases from 60.0 to 61.3; however, the difference in terms of precision and recall is impressive, both increasing with over 25%.

Table 1. Performance of all baseline and learning approaches, expressed in token accuracy, precision, recall, F-score, and WindowDiff. For WindowDiff, lower scores are better.

	Token	Segment			
	Acc.	Prec.	Rec.	$F_{\beta=1}$	WDiff
MajB	24.8	0.0	0.0	0.0	.346
ExactB	16.0	25.7	23.1	24.3	.425
UniB	27.0	8.9	22.8	12.8	.818
TriB	43.8	12.9	24.8	16.9	.582
TriB+Vote	45.1	14.9	27.8	19.4	.536
MBL rand.	44.6	7.1	19.2	10.4	.568
MBL bias	53.4	12.1	32.0	17.6	.533
HMM	60.0	47.1	43.9	45.4	.173
+ perfect seg.	61.3	74.0	69.2	71.5	.036

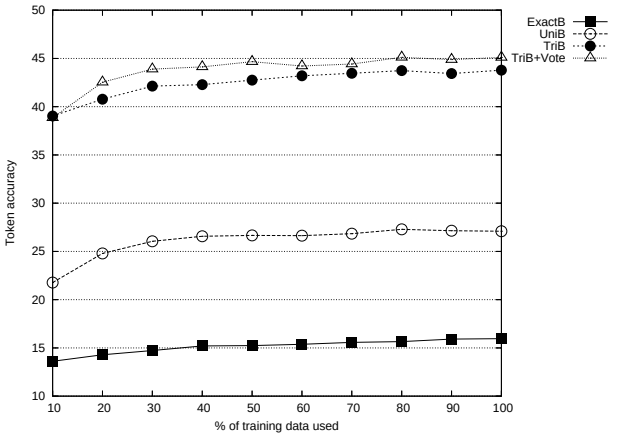

Figure 1. Token accuracies for the various baseline approaches trained on increasingly larger parts of the complete database (16,670 records).

4. Conclusion

Information extraction is often used to automate the process of filling a structured database with content extracted from written texts. Supervised machine learning approaches have been successfully applied for creating systems capable of performing this task. However, the supervised nature of these approaches requires large amounts of annotated training data; the acquisition of which is often a laborious and time-consuming process. In this study, we experimented with two machine learning techniques that do not require such annotated training data, but can be trained on a database containing information derived from the type of documents targeted by the application.

The first approach is an attempt to employ a standard supervised machine learning algorithm, training it on artificial labelled training data. These data are created by concatenating the contents of the cells of the database records in random order. Experiments with this approach pointed out that truly random concatenation of database fields results in weak performance; a rather simple baseline approach, which only matches substrings of a field book entry with the contents of the database, leads to better results. However, if a small amount of annotated field book entries is available —in this study, 10 entries turned out to be sufficient— one can estimate field ordering probabilities that can be used to generate more realistic training data from the

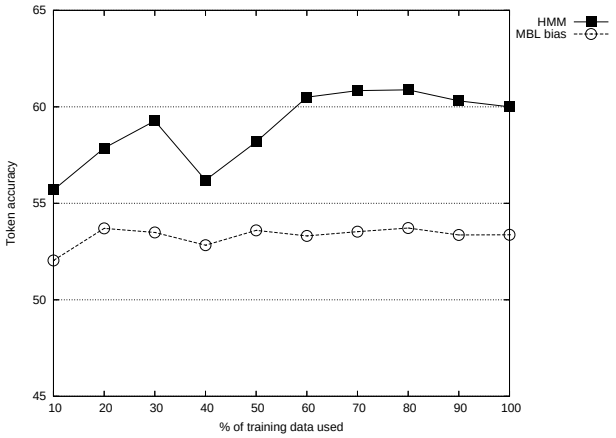

Figure 2. Token accuracies for the two machine learning approaches trained on increasingly larger parts of the complete database (16,670 records).

database. A machine learner trained on these data labelled 10% more tokens correctly than the system trained on the randomly generated data.

Our second approach is based on unsupervised hidden markov modelling. First, an n-gram language model is used to divide the field book entries into unlabelled segments. Then, a Hidden Markov Model is trained on these segmented entries using the Baum-Welch algorithm to estimate state-transition probabilities. The resulting HMM labels the segments found in the preceding segmentation step. The HMM state-emission distributions are estimated by training n-gram language models on the contents of the database columns.

The performance of the HMM proved to be superior to the other approaches, outperforming the supervised learner by labelling 61% of the tokens correctly, as well as attaining good results in terms of segment-level F-score. The weak part of the HMM approach turned out to be the initial segmentation. If this part is replaced by a perfect segmentation, F-score increases from 45% to 72%, which for many information extraction tasks is considered quite acceptable. Future work should therefore mainly focus on improving this initial segmentation step; preferably using no labelled training data but the contents of the specimen database.

Acknowledgments

The research reported in this paper was funded by NWO, the Netherlands Organisation for Scientific Research, as part of the IMIX and CATCH programmes. We would like to thank Antal van den Bosch for useful discussions, and Marieke van Erp for labelling part of the data.

References

Baum, L., Petrie, T., Soules, G., & Weiss, N. (1970). A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, *41*, 164–171.

Borkar, V., Deshmukh, K., & Sarawagi, S. (2001). Automatic segmentation of text into structured records. *Proc. 2001 ACM SIGMOD* (pp. 175–186).

Daelemans, W., Zavrel, J., Van der Sloot, K., & Van den Bosch, A. (2004). *TiMBL: Tilburg memory based learner, version 5.1, reference guide.* ILK Research Group Technical Report Series no. 04-02.

Freitag, D., & Kushmerick, N. (2000). Boosted wrapper induction. *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI/IAAI-2000)* (pp. 577–583).

Grenager, T., Klein, D., & Manning, C. D. (2005). Unsupervised learning of field segmentation models for information extraction. *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)* (pp. 371–378).

Kushmerick, N., Johnston, E., & McGuinness, S. (2001). Information extraction by text classification. *Proc. of the IJCAI-01 Workshop on Adaptive Text Extraction and Mining.*

Pevzner, L., & Hearst, M. (2002). A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, *28*, 19–36.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications inspeech recognition. *Proceedings of the IEEE*, *77*, 257–286.

Stolcke, A. (2002). SRILM - an extensible language modeling toolkit. *Proc. ICSLP*, *2*, 901–904.

Van den Bosch, A., & Daelemans, W. (2005). Improving sequence segmentation learning by predicting trigrams. *Proceedings of the Ninth Conference on Natural Language Learning, CoNLL-2005* (pp. 80–87).

Viola, P., & Narasimhand, M. (2005). Learning to extract information from semi-structured text using a discriminative context free grammar. *Proc. 28th ACM SIGIR* (pp. 330–337).