Information Extraction from Web Product Catalogues

Martin Labský and Vojtěch Svátek

Department of Information and Knowledge Engineering,
University of Economics, Prague, W. Churchill Sq. 4, 130 67 Praha 3, Czech Republic
{labsky,svatek}@vse.cz

Abstract. In this paper we present preliminary results for information extraction (IE) performed over a set of HTML documents using Hidden Markov Models (HMMs). In our experiments, we restrict ourselves to the domain of bike products sold on the Internet. The information to be extracted consists of bike model attributes and details regarding the company's offer. We experiment with three approaches utilising HMMs and present results in terms of precision and recall.

1 Introduction

Information extraction (IE) from the WWW is receiving much attention since it provides semantic interpretation for data found on web pages, allowing for the creation of semantics-aware search engines.

IE has been successfully applied to tasks such as collecting structured job advertisements, seminar announcements, addresses or publication records [1], [3]. In our work, we experiment with IE from *product catalogues* present on the WWW in the form of HTML documents. Our major aim is to build a domain-specific semantic search engine capable of answering queries e.g. about product names and prices and pointing the user to the original web sites. As an example domain, we chose bike products.

2 IE from Product Catalogues

The typical web IE tasks mentioned above are often limited to sets of pages with similar structure, design and content. In our work, we want to test the viability of using *Hidden Markov Models* (HMMs) for IE from very diverse product catalogues found all over the Internet.

HMMs are finite state machines augmented with state transition probabilities and lexical probability distributions for each state. Text is modelled as a sequence of tokens (in our case including words, punctuation and formatting symbols). When applying an HMM to text, the given sequence of tokens is assumed to have been generated by that model. Provided states of the model are associated with semantic slots (to be filled in with extracted text), we are interested in recovering the most probable state sequence that would have generated our text,

and thereby obtaining its most probable semantic interpretation. This task is effectively solved by the *Viterbi dynamic programming algorithm* [2].

To provide us with training and test data, we manually annotated 100 product catalogues randomly chosen from bike shop websites in the UK[1]. Each document contains from 1 to 50 bike offers. Some of the slots we use are shown in Table 1.

Before applying HMMs, we transformed each document into a sequence of HTML block elements (e.g. paragraphs, table cells) that directly contain potentially interesting data (in our case, any text or images). Furthermore, inline HTML tags (e.g. fonts) were substituted with abstract tag classes (e.g. ????).

3 Experiments and Results

Experiments were carried out using three different HMM architectures. In all cases, we experimented with a *trigram*[2] HMM instead of the commonly used bigram, seeking to capture farther-reaching dependencies between slots. The *smoothing* method utilised for lexical probabilities was absolute discounting similar to [3], and linear interpolation [2] for transitional probabilities.

3.1 The Naive Model

In the naive approach, we represent each semantic slot with a single *target* state. Additionally, we define a *prefix* and a *suffix* state for each slot, responsible for modelling typical left and right contexts. Finally we use a single *background* state producing uninteresting data. In contrary to [1], where independent models are built for each slot, we create a single model containing all slots with the hope of capturing dependencies between different slots (e.g. price typically following name).

We train the naive model directly using counts from labelled training data, as there always is a single state sequence visible in each labelled document. Since the prefix and suffix states for each slot are not directly labelled in the data, we treat k preceding and k following tokens as being emitted by these states (in our experiments $k = 2$).

3.2 Word N-gram Models

In the naive approach, the internal structures of a slot is ignored since we model each slot's content with a lexical distribution of a single state. However, incorporating structural information about slots leads to a more precise model and might help recognise slot values containing unseen tokens.

[1] The documents were picked from the Google Directory node *Sports-Cycling-Bike Shops-Europe-UK-England*; the labelled collection is available at *http://rainbow.vse.cz*

[2] I.e., with transition probabilities conditioned by *two* previous states.

In this approach, we substitute the unigram lexical distributions of chosen states with *n-gram* lexical distributions. The state structure and transition distributions remain unchanged compared to the naive model. In our experiments we use *word trigram*[3] models for selected slots, trained from the particular slot's training data and smoothed via linear interpolation with weights obtained using the EM algorithm.

To obtain the most probable state sequence $s(1), ...s(n)$ for observed tokens $w_1, ..., w_n$ within this model, a simple modification to the abovementioned Viterbi algorithm was designed. For simplicity, let us explain it for the case of bigram HMM only. $\gamma_{t,i}$ is a dynamic programming variable denoting the probability of the best state sequence leading to state s_i at time t. Normally, it is computed using just (1) and (2). In the proposed extension, word bigram probability is used instead of the unigram probability when the state considered as previous (s_k) is the same as the current state (s_i).

$$\gamma_{t,i} = max_k \ (\gamma_{t-1,k} P(s_i|s_k) Lex(w_t|s_i)) \tag{1}$$

where

$$Lex(w_t|s_i) = P(w_t|s_i) \ if(k \neq i) \tag{2}$$

$$Lex(w_t|s_i) = P(w_t|w_{t-1}, s_i) \ if(k = i) \tag{3}$$

The approach is analogous for a trigram HMM, where we may also use a word trigram model.

3.3 HMM Submodels

The third approach we experimented with is used e.g. in [1]. In this case, we learn an HMM submodel for each semantic slot having significant internal structure. HMM submodels are learnt using the unsupervised Baum-Welch algorithm [2] from the corresponding slot's data, with the desired number of states determined experimentally. In contrast to the naive model, the global trigram model structure remains the same, however the learnt submodels are used in place of the original singleton target states.

In Figure 1 we show an example of a three-state submodel trained for the bike name slot. Only transitions with probability higher than 0.05 and the most frequent words are shown for each state. It is interesting to note that the model learnt to recognise bike company names in state 1, bike model names in state 2, and generic properties such as colours, sizes or brake types in state 3.

4 Conclusion and Discussion

The results presented in Table 1 for the name and price slots were obtained using the naive, word n-gram and submodel approach respectively. The remaining slots

[3] I.e., with lexical probabilities conditioned by the words generated in *two* previous states (provided the slot remains the same).

Table 1. 10-fold cross-validation results

Slot	*Recall*	*Precision*	*# instances*
name	77.9 78.6 83.63	63.5 65.6 62.1	927
price	98.9 99.1 98.8	89.5 88.9 86.9	971
picture	69.0	89.6	359
speed	86.8	93.6	186
size	83.2	93.7	173
year	98.1	70.0	160

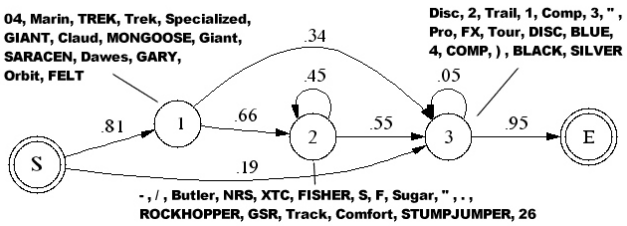

Fig. 1. 3-state submodel trained for bike name

do not exhibit significant internal structure and were thus modelled only with the naive approach, yielding almost identical results for all three models. Precision and recall was measured on a per-token basis. All results were obtained using 10-fold cross-validation on the whole set of labelled 100 documents, with the presented values averaged.

Our results confirm that incorporating structural information about slots might be significant for increasing both precision and recall. Both non-naive approaches to modelling slot values however suffer from data sparseness, which probably causes the degradation of precision for the bike name submodel.

The research is partially supported by grant no.201/03/1318 of the Grant Agency of the Czech Republic, "Intelligent analysis of the WWW content and structure."

References

1. A. McCallum, D. Freitag. Information extraction with hmms and shrinkage. *Proceedings of the AAAI-99 Workshop on Machine Learning for IE*, 1999.
2. F. Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts, 1997.
3. V. Borkar, K. Deshmukh, S. Sarawagi. Automatic segmentation of text into structured records. *SIGMOD Conference*, 2001.