Product information extraction from semistructured documents using HMMs

Martin Labský

Department of Information and Knowledge Engineering,
University of Economics, Prague, W. Churchill Sq. 4, 130 67 Praha 3, Czech Republic
labsky@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 a simple extension to HMMs which models the inner structure of chosen extracted slots. Results are presented in terms of precision and recall.

1 Introduction

Information extraction (IE) is a process of automatically annotating parts of text with semantic tags. Typically, these tags correspond to attributes of entities in some model of the domain (often a database schema or an ontology). In combination with such model, IE is often used for database (or knowledge base, KB) population tasks.

In our experiments, we work with a simple ontology[1] about bikes, shops and bike parts, and our goal is to automatically populate and update the ontology's KB with instances from the Internet. The desired application is a limited-scope semantic search engine[2], capable of answering structured queries about the instances present in its KB and pointing its users to the original company websites.

For this domain, we decided to use statistical methods for IE, since they generally tend to be robust when encountering biased and highly varying data. HMMs have been successfully used for IE (e.g. [1], [2]) since the end of the last decade. Typically, information has been extracted from data such as seminar invitations (speaker, time, place) or job advertisements. However, these methods have mostly been evaluated on similar documents (in terms of size, formatting, and amount of information presented), often obtained from a single source. In our work, we experiment with a more diverse set of documents which comes from arbitrary shops' websites.

HMMs are finite state machines augmented with state transition probabilities and lexical probability distributions for each state. Text is modelled as a sequence

[1] bike ontology is available at http://rainbow.vse.cz/bikes/

[2] the search engine will integrate other components of the Rainbow [4] framework

of tokens (may include words, punctuation, formatting symbols). When applying an HMM to text, the text's token sequence is assumed to have been generated by that model (by transitioning between states and emitting a token in each state). In the simplest form, each semantic tag is modelled by a single HMM state, plus there is a "background" state for modelling uninteresting tokens. Both the transition and lexical distributions of the model can be trained from labeled training data. During tagging, we are interested in the most likely state (tag) sequence of the model that would have generated the input text. This task is solved effectively by the Viterbi dynamic programming algorithm (e.g. [2]).

2 The Model Structure

In our approach, we use an HMM state structure inspired by [1]. The extracted tokens are modelled using the so-called *target* states, which may be accompanied by two types of states responsible for representing their characteristic context - the *prefix* and *suffix* states. Irrelevant words are modelled by a single *background* state. Contrary to [1], which uses independent HMMs trained for each tag, we train a composite HMM which contains several types of target states and is thus capable of tagging with multiple tags. This is an approach similar to [2] and for our task it is advantageous since it extensively exploits ordering relations between nearby tags (e.g., bike model name is often followed by its price). Compared to a single-tag model, this architecture also prevents crossing tags at the moment of merging results of individual models.

We model the whole document using a single HMM. In order to capture the ordering relationships between extracted elements, which often are not adjacent, we work with a tag trigram model (HMM state is represented by a pair of tags), which we expect to be capable of learning more distant relations than the typically used tag bigram model. Deleted interpolation [3] is used to smooth transition probabilities. To compute unknown word lexical probabilities for each tag, we use absolute discounting in the same way as [2].

3 Training the Model

According to attributes in the bike ontology, we manually annotated[3] a set of 100 HTML documents obtained from about 40 bike shop websites in the UK. For choosing these websites, we used the first 40 addresses from the Google Directory *Sports-Cycling-Bike Shops-Europe-UK-England*. From each such website we manually selected 1 to 5 documents offering at least one bike model and containing at least its name and price. The annotations have been made by inserting SGML tags into the raw downloaded documents. These documents are highly diverse, offering from 1 to about 50 bikes each, with different levels of detail of the bikes' descriptions.

[3] training data is available under http://rainbow.vse.cz/bikes/

Before training and tagging, we employ extensive preprocessing, which converts HTML documents into suitable sequences of tokens. For the most part, this step disposes of a part of HTML formatting tags, canonises HTML entities, detects word boundaries, and substitutes certain tokens with token classes. For example, all HTML tags denoting blocks are classified as `<block>`. Similar classification is done for inline HTML tags, images and forms. Since we are interested in extracting only words and images, and the HMM has limited memory, we construct the token sequence by concatenating contents of only those HTML blocks that directly (i.e. not only via other blocks) contain words or images. Finally, we treat all numbers as a single class, as well as all words containing both digits and letters.

The preprocessed token sequences are used to train the transition and lexical probability distributions using ratios of counts, since in our approach there is always a single state sequence visible in a given training document. In the training data, we observe just the labeled extracted tags. To populate also the prefix and suffix states, we consider the 2 preceding and the 2 following tokens as members of the prefix and suffix states, respectively. We use less then 2 token windows in cases of encountering another nearby semantic tag or document boundaries.

For modelling the distributions of background tokens and each target's prefix and suffix tokens, we use a naive approach and model each of these using a single lexical distribution, thereby ignoring their internal structures. However, for certain tags such as *name* and *price*, there is a significant structure that needs to be modelled. One approach is to model the tag's content by a small HMM that is effectively used in place of the original single target state ([1], [2]). We experimented with a different approach. We conditioned the state's lexical probability not only on its tag, but also on a short history of tokens seen so far within the same tag. In this way, the lexical distribution of a state having tag t is made dependent on the previous n-word history provided these n words are tagged with t. We speculate this might make tagging more exact in case enough training data is provided. In the Viterbi algorithm, the tag trigram model enables us to look at a history of 2 words, knowing the tags assigned to them. Therefore, we are able to use a word bigram and/or trigram model as the basis for computing the target state's lexical distribution. Specifically, we interpolate the original word unigram distribution used for a particular tag with a word bigram distribution, and eventually with a word trigram distribution. Weights of these distributions are computed using the EM algorithm [1] over the tag's training data. Our experiments show a slight increase in both precision and recall for most tags modelled in this way.

4 Tagging Results

The tagging results have been evaluated on test data using precision and recall on a per-token basis. For each word, the correct and assigned tags have been compared. In Table 1, we present results for selected attributes of offered bike models. Results of modelling tags with word bi- and trigram models are brack-

eted. All results were obtained using 10-fold cross-validation on the whole set of labeled 100 documents, with the presented values averaged.

Table 1. 10-fold cross-validation results for selected tags

Tag	recall	precision	instances
name	77.9 (78.6)	63.5 (65.6)	927
price	98.9 (99.1)	89.5 (88.9)	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

5 Conclusions and Future Work

We have presented preliminary results of product IE over a set of diverse documents from multiple sources. There are several directions for future work. After we improve our results, we need to construct instances (template IE) by combining annotations into structures according to the ontology's classes and constraints (e.g. cardinality, data types or axioms). A way of modifying the Viterbi algorithm to support constraints is described in [2].

In our experiments with product catalogues, we have noticed that the tagger often classifies most product entries correctly, but occasionally misses several product names, because they are very different from the training data. We developed a simple symbolic algorithm which identifies similar structural patterns in a document. For example, the HTML tag sequence `<td> <a> <font> <br/> </font> </a> </td>` with arbitrary words in between appears 34 times in one of our training documents. In this case, the tagger successfully annotates 28 product names contained in these patterns between `<font>` and `<br/>`, but misses the remaining 6. In cases like this, we want to recover the remaining product names and use them to enrich the model's training data. By learning novel product names from these "easy" pages, the model will learn to recognise them also in less structured documents. Other bootstrapping strategies are covered in [5].

References

1. Freitag D., McCallum A.: Information extraction with HMMs and shrinkage. In: Proceedings of the AAAI-99 Workshop on Machine Learning for IE, 1999.
2. Borkar V., Deshmukh K., Sarawagi S.: Automatic segmentation of text into structured records. In: SIGMOD Conference, 2001.
3. Schroeder I: A Case Study in Part-of-Speech Tagging Using the ICOPOST Toolkit. University of Hamburg, Computer Science Department, NATS, 2002.
4. Svatek V. et al: Rainbow - Multiway Semantic Analysis of Websites. In: The 2nd Int. DEXA Workshop on Web Semantics, IEEE Computer Society Press 2003.
5. Dingli A., Ciravegna F., Guthrie D., Wilks Y.: Mining Web Sites Using Unsupervised Adaptive Information Extraction. In: EACL, 2003.