Information Extraction by Convergent Boundary Classification

Aidan Finn and Nicholas Kushmerick
Smart Media Institute, Department of Computer Science, University College Dublin
{aidan.finn, nick}@ucd.ie

Abstract

We investigate the application of classification techniques to the problem of information extraction (IE). In particular we use support vector machines and several different feature-sets to build a set of classifiers for information extraction. We show that this approach is competitive with current state-of-the-art information extraction algorithms based on specialized learning algorithms. We also introduce a new technique for improving the recall of IE systems called convergent boundary classification. We show that this can give significant improvement in the performance of our IE system and gives a system with both high precision and high recall.

Introduction

Information extraction (IE) is the process of identifying a set of pre-defined relevant items in text documents. Numerous IE systems based on machine learning techniques have been proposed recently. Many of these algorithms are "monolithic" in the sense that there is no clean separation between the learning algorithm and the features used for learning. Furthermore, many of the proposed algorithms effectively reinvent some aspects of machine learning rather than exploit existing machine learning algorithms.

In this paper, we investigate how relatively "standard" machine learning techniques can be applied to information extraction. We adopt the standard "IE as classification" formalization (Freitag & Kushmerick 2000; Ciravegna 2001), in which IE becomes the task of classifying every document position as either the start of a field to extract, the end of a field, or neither. We investigate how different feature-sets contribute to the performance of our system.

Based on these initial results, we then describe improvements on this basic approach that give higher performance on a variety of benchmark IE tasks. Our enhancements–which we call convergent boundary classification–consist of combining the predictions of two sets of classifiers, one set with high precision and one with high recall.

The intuition behind this approach is as follows: We assume the base classifiers have high precision and predict very few false positives. To extract a fragment we need to identify both its start and end tags. If the base classifier predicts one tag (start or end) but not the other, we assume that it is correct. We use this prediction as a guide to a second classifier to identify the missing complementary tag.

We make two contributions. First, we show that the use of an off-the-shelf support vector machine implementation is competitive with current IE systems based on specialized learning algorithms. Second, we introduce a novel multi-level approach called convergent boundary classification and demonstrate that this new approach outperforms current systems.

Previous algorithms

We begin with a discussion and comparison of some of the more prominent adaptive IE algorithms.

Rapier (Califf & Mooney 1999) uses inductive logic programming techniques to discover rules for extracting fields from documents. It does not try to identify start and end tags separately, but learns to identify relevant strings in their entirety. Rapier performs specific-to-general bottom-up search by starting with the most specific rule for each positive training example and repeatedly trying to generalize these rules to cover more positive examples. Rapier uses as its features the tokens, part-of-speech information and some semantic class information.

BWI (Freitag & Kushmerick 2000) learns a large number of simple wrapper patterns, and combines them using boosting. BWI learns separate models for identifying start and end tags and then uses a histogram of training fragment lengths to estimate the accuracy of pairing a given start and end tag. BWI learns to identify start and end tags using a form of specific-to-general search. BWI's features consist of the actual tokens, supplemented by a number of orthographic generalizations (alphabetic, capitalized, alphanumeric, lower-case, numeric, punctuation), as well as a modest amount of lexical knowledge (a list of first and last names).

LP^2 (Ciravegna 2001) learns symbolic rules for identifying start and end tags. Like BWI, it identifies the starts and ends of fields separately. In addition to token and orthographic features, LP^2 uses some shallow linguistic information such as morphological and part-of-speech information. It also uses a user-defined dictionary or gazetteer. Its learning algorithm is a covering algorithm which starts with specific rules and tries to generalize them to cover as many positive examples as possible. This process is supplemented by learning correction rules that shift predicted tags in order to correct some errors that the learner makes.

BIEN (Peshkin & Pfeffer 2003) uses a dynamic Bayesian network for learning. It uses several different kinds of features: in addition to the token information and POS information, it uses chunking, a gazetteer, a lemmatiser and semantic and orthographic features.

BWI uses the fewest features: it uses just the tokens and some orthographic information. LP^2 and RAPIER supplement these features with part-of-speech and semantic information. BIEN has the most sophisticated feature set.

It is difficult to compare each of these systems directly for several reasons. First, there are a variety of "simple" methodological differences (e.g., using a slightly different scoring mechanism) (Lavelli *et al.* 2004). More significantly, most of the literature reports only results for the system in its entirety. To summarize, it is difficult to know how much of the performance differences arise from different feature sets, how much from different learning algorithms, and how much from differences in experimental procedure.

The ELIE algorithm

Information Extraction as classification

Following (Freitag & Kushmerick 2000; Ciravegna 2001), the approach that we use is to treat the identification of fragment start and end positions as distinct token classification tasks. The instances are all tokens in the document. All tokens that begin a labeled field are positive instances for the start classifier, while all the other tokens become negative instances for this classifier. Similarly, the positive examples for the end classifier are the last tokens of each labeled field, and the other instances are negative examples.

Each instance has a set of features that describe the given token. The features include the specific token, as well as part-of-speech (POS), chunking, orthographic and gazetteer information. The features are described in more detail below. In addition, we add features to represent a fixed-width window of tokens on either side of the instance's token.

Features and encoding

ELIE uses several kinds of features.

Token The actual token.

POS The part-of-speech of the token. Each token is tagged with its corresponding POS using Brill's POS tagger (Brill 1994). We also represent chunking information about the tokens. The POS tags are grouped into noun-phrases and verb-phrases.

Gaz The values associated with the token in a gazetteer. The gazetteer is a user-defined dictionary. It contains lists of first-names and last-names taken from the U.S. census bureau, a list of countries and cities, time identifiers (am, pm), titles (Jr., Mr), and a list of location identifiers used by the U.S. postal service (street, boulevard).

Orthographic These features give various orthographic information about the token. Examples of these features include whether the token is upper-case, lower-case, capitalized, alphabetic, numeric or punctuation.

To represent an instance, we encode all these features for that particular token. In addition, for a fixed window size of w, we add the same features for the previous w tokens and the next w tokens. For example, the string

```
Place:     WeH 4601
Speaker:   Alex Pentland
```

would be tokenized and tagged as

Token	POS/Chunk	Gaz	Orthographic
place	NNP		alpha, cap
:			punct
weh	NNP		alpha, cap
4601			num
\n			
speaker	NNP		alpha, cap
:			punct
alex	NNP, NP_s	firstname	alpha, cap
pentland	NNP, NP_e	lastname	alpha, cap

If we encode the instance centered at the token 'alex', using a window of size 1, we would get the following features:

```
T_alex_0, T_:-1, T_pentland_+1,
P_NNP_0, P_NNP_+1,
C_NP_s_0, C_NP_e_+1,
G_firstname_0, G_lastname_+1,
O_alpha_0, O_cap_0 , O_punct_-1,
O_alpha_+1, O_cap_+1
```

All tokens in the dataset are encoded in this manner. This gives a very large number of attributes. We therefore filter the attributes according to information gain (Quinlan 1993) in order to discard irrelevant features and reduce learning time.

Learning with ELIE

The ELIE algorithm has two distinct phases. In the first phase, ELIE simply learns to detect the start and end of fragments to be extracted. Our experiments demonstrate that this first phase generally has high precision but low recall. The second phase is designed to increase recall. We find that very often false negatives are "almost" extracted (the start but not the end is correctly identified, or the end but not the start). In the second phase, which we call convergent boundary classification, ELIE is trained to detect either the end of a fragment given its beginning, or the beginning of a fragment given its end. In this way, ELIE iteratively converges to the correct boundary classifications.

Level One (L1) learning. The L1 learner treats IE as a standard classification task, augmented with a simple mechanism to attach predicted start and end tags.

Figure 1 show the learning process. The set of training examples are converted to a set of instances for the start and end tags as described above. Each token in each training

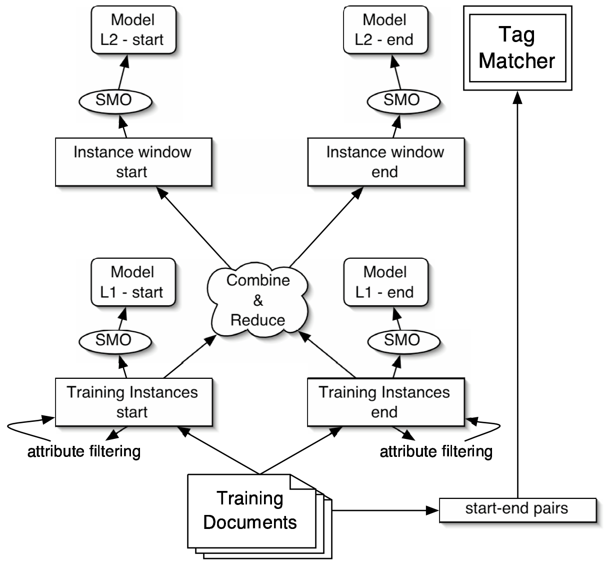

Figure 1: Learning

document becomes a single instance, and is either a positive or negative example of a start or end tag. Each of these instances is encoded according to several features for the particular token in question and the tokens surrounding it. Then the attributes are filtered according to information gain. These instances are passed to a learning algorithm[1] which uses them to learn a model. At the end of the L1 training phase we have models for start and end tags and all the start-end pairs.

The start-end pairs are passed to the tag-matcher which is charged with matching start and end tags. Our experiments involve a tag-matcher which generates a histogram based on the number of tokens between each start and end tag in the training data. When matching predictions, the probability of a start-tag being paired with an end-tag is estimated as the proportion with which a field of that length occurred in the training data. This approach performs adequately and we don't focus on tag-matching further in this paper. A more intelligent tag-matcher may improve performance in the future. For example, the tag-matcher might incorporate a learning component that learns to shift tags and correct errors in the output predictions.

Level two (L2) learning The L1 learner builds its model based on a very large number of negative instances and a small number of positive instances. Therefore the prior probability that an arbitrary instance is a boundary is very small. This gives a model that has very high precision. Because the prior probability of predicting a tag is so low, when we actual do predict a tag, it is very likely that the prediction is correct. The L1 model is therefore much more likely to produce false negatives than false positives.

[1]Our current experiments are based on Weka (Witten & Frank 2000). We used Weka's SMO (Platt 1998; Keerthi *et al.* 2001) algorithm for the learner, but other algorithms could be substituted. We performed some initial experiments using various learning algorithms, and SMO was consistently one of the most competitive.

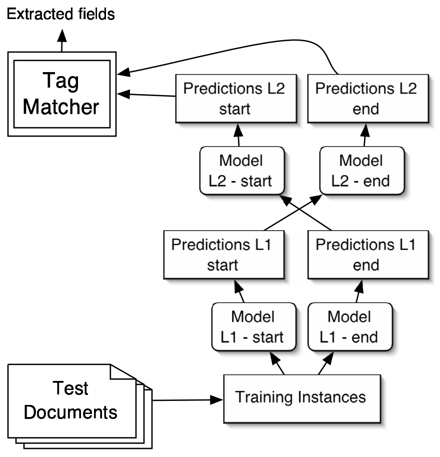

Figure 2: Extracting

The L2 learner is learned from training data in which the prior probability that a given instance is a boundary is much higher than for the L1 learner. This "focused" training data is constructed as follows. When building the L2 start model, we take only the instances that occur a fixed distance before an end tag. Similarly, for the L2 end model, we use only instances that occur a fixed distance after a start tag.

For example, an L2 window of size 10 means that the L2 start model is built using only all the groups of 10 instances that occur before an end-tag in the training data, while the L2 end model is built using only those instances that occur in the 10 instances after a start tag in the training data. Note that these L2 instances are encoded in the same way as for L1; the difference is simply that the L2 learner is only allowed to look at a small subset of the available training data.

This technique for selecting training data means that the L2 models are likely to have much higher recall but lower precision. If we were to blindly apply the L2 model to the entire document, it would generate a lot of false positives. Therefore, as shown in Figure 2, the reason we can use the L2 model to improve performance is that we only apply it to regions of documents where the L1 model has made a prediction. Specifically, during extraction, the L2 classifiers use the predictions of the L1 models to identify parts of the document that are predicted to contain fields. Since the L1 classifiers generally have high precision but low recall, the intent is that this procedure will enable ELIE to iteratively converge to the correct boundary classifications.

Experiments

We evaluated our ELIE algorithm on two benchmark datasets and compared the results to those achieved by other IE systems.

Evaluation method A truly comprehensive comparison would compare each algorithm on the same dataset, using the same splits, and the exact same scoring system. Unfortunately, a conclusive comparison of the different IE systems is impossible using the current published results. The other systems are evaluated using slightly different methodologies (Lavelli *et al.* 2004).

There are several orthogonal issues regarding evaluation. The first is whether to give credit for partial matches (where an extracted fragment is correct at one end, but the other end is the wrong token). We take the more conservative approach of using exact matching only. Thus if the speaker is “Dr Lee, PhD” and we extract only “Dr Lee”, this would count as both a false positive and a false negative. Our evaluation is conservative with respect to counting true positives: we must identify both the start and end exactly.

The second issue is how to count correct extractions and errors. The most conservative approach is to require the system to extract all occurrences of a field (all slot occurrences: ASO) to get full credit. Thus if a document contained a target field which had two occurrences, “2pm” and “2:00”, then the system would be required to extract both.

An alternative is a “template” approach (single-slot-occurrence: SSO). In this case it is sufficient to extract either 2pm or 2:00 as they refer to the same entity. It is assumed that there is one correct answer per slot, and the extraction algorithm’s most confident prediction is used as its prediction. SSO evaluation makes sense for fields where we know that they refer to a single value (e.g. the time that a seminar starts).

All the systems we compare to use some form of the template filling (SSO) results although they don’t specify exactly how they measure performance. Freitag, for example, assumes one filler per slot and discards all but the most confident predictions.

Experimental setup

We evaluate our system using two standard benchmark datasets: the seminar announcements (SA) dataset (Freitag 1998) and the job postings dataset (Califf & Mooney 1999). The SA dataset consists of 485 seminar announcements from Carnegie Mellon University detailing upcoming seminars. Each seminar is annotated with fields speaker, location, start-time and end-time. The jobs dataset consists of 300 newsgroup messaged detailing jobs available in the Austin area. The dataset has been annotated for 17 fields (see Table 3).

Both of these datasets contain numerous labeling errors and inconsistencies and therefore it is unlikely that any system can achieve perfect performance on them. In many documents some occurrences of a field are labeled while others are not. This affects the all-slots evaluation more that the template evaluation.

We used a 50:50 split of the dataset repeated 10 times. Results for BWI, RAPIER and LP^2 come from other sources (Lavelli *et al.* 2004), and where possible we use the same splits for each system.

All experiments use a window of length 3 tokens, and L2 lookahead/lookback of 10 tokens. On the SA dataset with a 50/50 split, this typically gives a set of approximately 80 thousand training instances, a few hundred of which are positive, and approximately 50 thousand attributes. These experiments have all features enabled initially, and then the top 5000 features ranked by information gain are used for learning the model.

	ELIE$_{L1}$			ELIE$_{L2}$			LP^2		
	P	R	F1	P	R	F1	P	R	F1
speaker	93.3	69.4	79.5	84.6	85.2	**84.9**	71.4	68.7	70.0
location	94.9	77.7	85.4	90.0	82.2	**85.9**	87.2	68.3	76.6
stime	96.7	85.2	**90.6**	84.7	96.3	90.2	89.0	87.7	88.3
etime	97.6	65.8	78.5	94.8	94.4	**94.6**	95.4	86.5	90.8

Table 1: Comparing ELIE and LP^2 on the Seminar Announcements dataset using ASO evaluation

	ELIE$_{L1}$			ELIE$_{L2}$			BWI			LP^2				Rapier	
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
speaker	95.8	76.2	84.9	91.0	86.0	**88.5**	79.1	59.2	67.7	87.0	70.0	77.6	80.9	39.4	53.0
location	96.1	75.9	84.8	93.1	80.7	**86.5**	85.4	69.6	76.7	87.0	66.0	75.1	91.0	60.5	72.7
stime	99.0	94.4	96.6	98.6	98.5	98.5	99.6	99.6	**99.6**	99.0	99.0	99.0	96.5	95.3	95.9
etime	99.0	77.8	87.0	95.7	97.3	**96.4**	94.4	94.4	94.4	94.0	97.0	95.5	95.8	96.6	96.2

Table 2: Comparing ELIE to other IE systems on the Seminar Announcements Dataset

Experimental results

Figure 1 compares ELIE and LP^2 using ASO evaluation[2]. We show results for our system with and without L2 classification. The L1 results are measured by passing the predictions of the L1 classifiers directly to the Tag-Matcher, bypassing the L2 classifiers (see Figure2). Using this mode of evaluation, ELIE outperforms LP^2 on all fields. On three of the four fields ELIE$_{L2}$ performs better than ELIE$_{L1}$. On the speaker and etime fields, using the L2 classifier provides a large increase in performance. In these cases, precision drops, but recall increases significantly.

Figure 2 shows results of our system on the SA task using SSO evaluation. We compare these scores against published results for BWI, RAPIER and LP^2 which use some variant of SSO evaluation. In this case BWI slightly outperforms ELIE on the stime field, but ELIE$_{L2}$ performs best on the other three fields.

For all fields on this dataset, the L1 classifier has high precision but low recall. The L2 classifier always has significantly higher recall while maintaining high precision. In each case recall rises with only a small drop in precision and a higher F1 with L2 classification.

Peskin reports results for BIEN on the SA dataset of 76.9, 87.1, 96.0 and 98.8 for speaker, location, stime and etime. These results are using repeated 80:20 test:train splits. Our results are competitive with these (better on two fields and slightly worse on two fields) even though we used only 50% of the data for training. It is likely that our systems performance would improve further if we used 80% of the dataset for training.

Figure 3 shows the performance of ELIE on the jobs dataset. Because many of these fields can have multiple values, we report results for ELIE using ASO evaluation. However it is likely that the other systems may have used some form of SSO evaluation. On a majority of fields ELIE outperforms both LP^2 and RAPIER.

[2]These results were provided by Fabio Ciravegna.

	ELIE$_{L1}$			ELIE$_{L2}$			LP^2			Rapier		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
id	100	99.5	99.7	100	99.5	99.7	100.0	100.0	**100.0**	98.0	97.0	97.5
title	69.3	40.1	50.7	57.2	54.6	**55.8**	54.0	37.0	43.9	67.0	29.0	40.5
company	94.2	69.6	**79.8**	90.1	71.5	79.5	79.0	76.0	71.9	76.0	64.8	70.0
salary	73.9	52.8	61.4	71.5	62.1	66.3	77.0	53.0	62.8	89.2	54.2	**67.4**
recruiter	88.3	75.6	81.3	87.0	77.7	**82.0**	87.0	75.0	80.6	87.7	56.0	68.4
state	93.7	92.2	**92.9**	92.4	93.1	92.7	80.0	90.0	84.7	93.5	87.1	90.2
city	96.0	93.2	94.6	95.2	94.9	**95.1**	92.0	94.0	93.0	97.4	84.3	90.4
country	97.7	93.9	95.8	97.4	94.3	**95.8**	70.0	96.0	81.0	92.2	94.2	93.2
language	93.6	87.2	90.2	93.3	89.5	**91.4**	92.0	90.0	91.0	95.3	71.6	80.6
platform	87.1	64.6	74.1	84.8	75.5	79.8	81.0	80.0	**80.5**	92.2	59.7	72.5
application	85.5	51.2	63.9	81.0	61.4	69.7	86.0	72.0	**78.0**	87.5	57.4	69.3
area	72.5	29.0	41.4	61.8	40.3	48.7	70.0	64.0	**66.9**	66.6	31.1	42.4
req_years_ex	89.0	69.0	77.5	80.8	79.6	**80.0**	79.0	61.0	68.8	80.7	57.5	67.1
des_years_ex	96.5	70.0	80.8	93.1	75.4	82.9	67.0	55.0	60.4	94.6	81.4	**87.5**
req_degree	90.2	65.3	75.0	84.8	75.0	79.0	90.0	80.0	**84.7**	88.0	75.9	81.5
des_degree	95.5	45.1	58.9	67.8	50.8	55.2	90.0	51.0	65.1	86.7	61.9	**72.2**
post date	95.2	99.0	97.0	95.2	100.0	97.5	99.0	100.0	99.5	99.3	99.7	99.5

Table 3: Comparing ELIE to other IE systems on the Jobs Dataset

One some of these fields, L2 provides no improvement in performance over L1. Many of the fields in the jobs dataset are very short (e.g. city, state, country) and simply involve learning the token that occurs e.g. state is often 'TX'. In this situation there is little scope for generalization so the L1 classifier does as well as the L2 classifier.

In all cases, the L2 classifier has higher recall than the L1 classifier, and in almost all cases higher F1. Apart from the very short fields the improvement for L2 is large.

Feature experiment

To test how each set of features contributes to performance, we re-ran the experiment shown in Table 1 with various reduced sets of features.

This showed that for the location, stime and etime fields most of the performance comes using the token features alone. For the stime field we get F1=90.2% using all feature but we get F1=89.1% using tokens only. Similarly, for the etime and location fields we have 92.3% and 82.2% using only token features. This compares with 94.6% and 85.9% using all features. So the addition of the extra features does not contribute much to the performance on these fields.

For the speaker field, we get F1=65.0% using only the token features, compared to 84.9% using all features. Using the token and POS features gives F1=71.7%, using the token and orthographic features gives 72.3%, while the token and Gaz features gives 84%. We conclude that performance on the speaker field is greatly enhanced by the use of a gazetteer which can tag people's first and last names, and also by orthographic and POS features.

Discussion and summary

We compared our system to three others on the SA dataset and two others on the jobs dataset. On each dataset ELIE performed best of all the algorithms.

On the Seminar Announcements dataset, ELIE comprehensively outperforms LP^2, BWI and RAPIER. On the jobs dataset it outperforms both LP^2 and RAPIER on a majority of fields. In particular, on both datasets, ELIE performs significantly better on the fields that are longer and regarded as more difficult.

The L2 learner consistently improves recall while keeping precision high. On longer fields where the chance of boundary imprecision is higher the improvements are generally larger. The L2 classifier always improves recall and usually keeps precision high enough to improve F1.

An investigation of the errors that ELIE produces reveals that most errors are false negatives. Those that are false positives are mostly of two kinds. The first are as a result of using exact matching for evaluation, where we have tagged one end of the field correctly but not the other. The second occur as a result of labeling errors on the data where we extract something that should have been labeled but was not.

Conclusion

We have described an approach that treats information extraction as a token classification task. Using SMO, a fast support vector machine implementation, our ELIE system learns a set of classifiers for information extraction that are competitive with, and in many cases outperform, current IE systems.

We also described convergent boundary classification: a new way of combining classifiers for Information Extraction that yields significant performance improvements. This approach exploits the high precision of token classifiers to increase the recall of the IE system. Our system outperformed current IE systems on two benchmark datasets. On several fields, especially those that are longer and more difficult, it gave large improvements in performance.

Future work

There is plenty of scope for improvement in ELIE. We plan to analyze in detail why the L2 approach can give such dramatic improvements in recall, and specify precisely what properties of the system facilitate this.

Further experiments will involve testing the system on more datasets, testing more rigorously the influence of the various feature-sets on performance, and a comparison of different learning algorithms on this task independent of all other system variables.

Other learning components may improve ELIE's performance further, e.g. a component that learns to recognize and correct prediction errors similar to LP^2's correction rules. Another modification might add a third level classifier that takes the predictions of L1 and L2 and classifies the extracted fragment as being correct or not.

Finally, the tag-matcher currently extracts all possible start-end pairs that are of a length that occurred in the training corpus. This includes overlapping extractions. A more discerning tag-matcher may perform better.

Acknowledgments This research was funded by Science Foundation Ireland and the US Office of Naval Research. Thanks to Fabio Ciravegna and Dayne Freitag for their feedback and access to their results and systems.

References

Brill, E. 1994. Some advances in transformation-based parts of speech tagging. In *AAAI*.

Califf, M., and Mooney, R. 1999. Relational learning of pattern-match rules for information extraction. In *Proc. 16th Nat. Conf. Artifical Intelligence*.

Ciravegna, F. 2001. Adaptive information extraction from text by rule induction and generalisation. In *Proc. 17th Int. Joint Conf. Artificial Intelligence*.

Freitag, D., and Kushmerick, N. 2000. Boosted wrapper induction. In *Proc. 17th Nat. Conf. Artificial Intelligence*.

Freitag, D. 1998. *Machine Learning for Information Extraction in Informal Domains*. Ph.D. Dissertation, Carnegie Mellon University.

Keerthi, S.; Shevade, S.; Bhattacharyya, C.; and Murthy, K. 2001. Improvements to platt's smo algorithm for svm classifier design. In *Neural Computation*, volume 13:3.

Lavelli, A.; Califf, M. E.; Ciravegna, F.; Freitag, D.; Giuliano, C.; Kushmerick, N.; and Romano, L. 2004. A critical survey of the methodology for ie evaluation. In *4th International Conference on Language Resources and Evaluation*.

Peshkin, L., and Pfeffer, A. 2003. Baysian information extraction network. In *Proc.18th Int. Joint Conf. Artifical Intelligence*.

Platt, J. 1998. Fast training of support vector machines using sequential minimal optimization. In Schlkopf, B.; Burges, C.; and Smola, A., eds., *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Quinlan, R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufman.

Witten, I. H., and Frank, E. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.