Multi-level Boundary Classification for Information Extraction

Aidan Finn and Nicholas Kushmerick

Smart Media Institute, Computer Science Department, University College Dublin, Ireland
{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 IE. We show that this approach is competitive with current state-of-the-art IE algorithms based on specialized learning algorithms. We also introduce a new technique for improving the recall of our IE algorithm. This approach uses a two-level ensemble of classifiers to improve the recall of the extracted fragments while maintaining high precision. We show that this approach outperforms current state-of-the-art IE algorithms on several benchmark IE tasks.

1 Introduction

Information extraction (IE) is the process of identifying a set of pre-defined relevant items in text documents. Numerous IE algorithms 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, using their own specialized learning algorithms, 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 [6, 3], 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 algorithm. We show that this approach with support vector machine classification is competitive and in many cases superior to current state of the art approaches based on algorithms crafted specifically for IE.

Based on these initial results, we then describe improvements on this basic approach that give superior performance on a variety of benchmark IE tasks. Our enhancements—which we call multi-level 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. Our system consists of two sets of classifiers (L1 and L2). The L1 classifiers adopt the the standard "IE as classification" approach. L2 uses a second level of "biased" classifiers. To extract a fragment we need to identify both its start and end. If the L1 classifier predicts one end of the fragment (either the start or the end, but not both) we assume that it is correct. We use this prediction as a guide to the L2 classifier to identify the complete fragment (see Fig. 1).

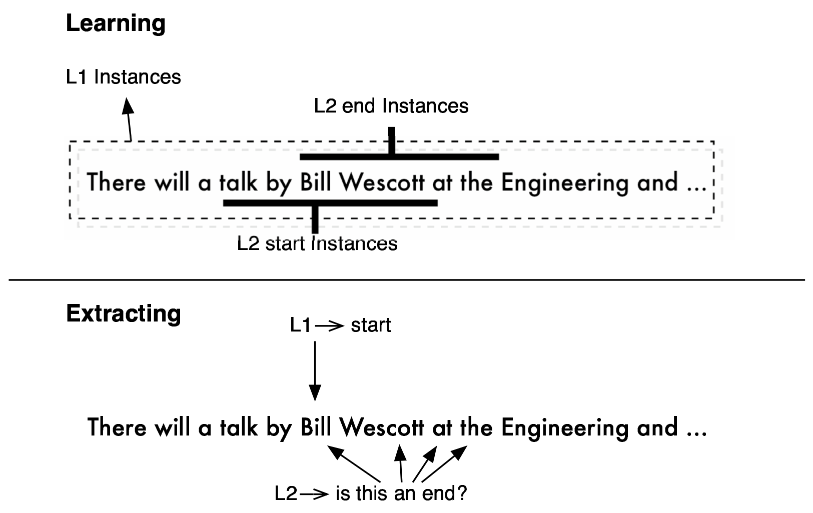
Learning
L1 Instances
L2 end Instances
There will a talk by Bill Wescott at the Engineering and ...
L2 start Instances
Extracting
L1 -> start
There will a talk by Bill Wescott at the Engineering and ...
L2 -> is this an end?

Fig. 1. L1 and L2: An example

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

2 Prior research

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

RAPIER [2] 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 [6] 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 [3] 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.

SNoW-IE [13]: SNoW [12] is a relational learning algorithm that is specifically tailored towards large scale learning tasks such as IE. SNoW-IE identifies fragments to be extracted rather than separately identifying start and end tags. It uses token, orthographic, POS and semantic features. It learns in two stages. The first stage involves filtering all the candidate fragments while the second involves picking the correct fragments from those remaining.

BWI uses the fewest features: it uses just the tokens and some orthographic information. LP^2 ,RAPIER and SNoW-IE supplement these features with part-of-speech and semantic information.

3 The ELIE algorithm

Information Extraction as classification. We treat tasks with multiple fields as multiple independent single-field extraction tasks i.e. we only extract one field at a time. Following [6, 3], we treat the identification of field start and end positions as distinct token classification tasks. 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.

Features and encoding. 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.

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 [1]. 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.

Encoding all tokens in the dataset in this manner gives a very large number of attributes. We therefore filter the attributes according to information gain [11] in order to discard irrelevant features and reduce learning time.

Relational information is encoded using additional features. 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, if we use a window size of 1, then each instance has a feature to represent the token for that instance, the token of the preceding instance and the token of the next instance. Similarly, there are features to represent the POS, gaz and orthographic information of the current instance and the previous and next instances.

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 ELIE is trained to detect either the end of a fragment given its beginning, or the beginning of a fragment given its end.

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.

Fig. 2 shows 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 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 derives 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 the 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.

[1] Our current experiments are based on Weka [14]. We used Weka's SMO [10] algorithm for the learner, but other algorithms could be substituted.

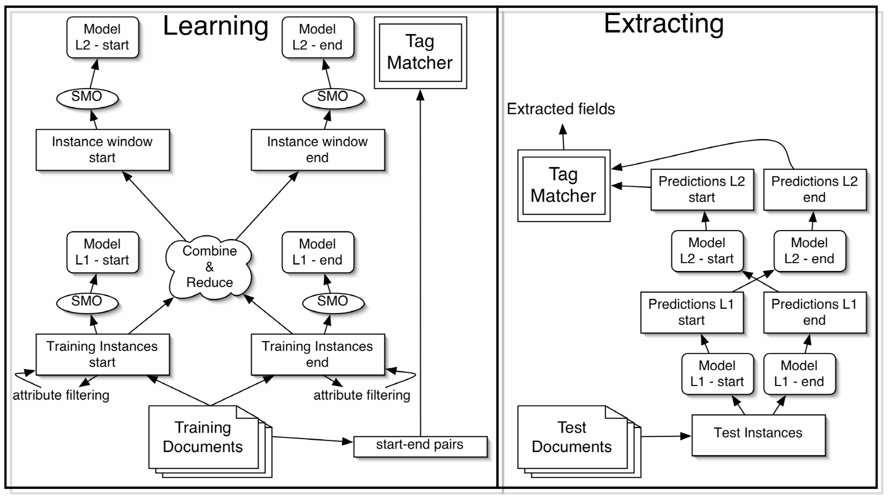

Fig. 2. ELIE architecture.

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 very low prior probability means that the L1 model is much more likely to produce false negatives than false positives.

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 and the number of irrelevant instances is vastly reduced. 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.

Fig. 1 shows an example of the the instances used by L1 and L2 with a lookahead/lookback of 3. In this example the token "Bill" is the start of a field and the token "Wescott" is the end of a field. When building the L1 classifiers we use all the available instances. When building the L2 start model we use the end token and the 3 tokens preceding it. When building the end model we use the start token and the three tokens following it. 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. When extracting, the L2 end classifier is only applied to the three tokens following the token which L1 tagged as a start and the token itself. Similarly the L2 start classifier is only applied to instances tagged as an end by L1 and the three preceding tokens.

This technique for selecting training data means that the L2 models are likely to have much higher recall but lower precision than L1 models. 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 Fig. 2 and Fig. 1, 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 converge to the correct boundary classifications.

The two level approach takes advantage of the fact that at L1 we have two highly dependent learners, each with very high precision. Thus a prediction by one of them indicates with very high probability that the second should make a prediction. When training the L2 classifier, we drastically alter the prior probabilities of the training data by using only the instances within a fixed distance before or after an annotated start or end. This L2 classifier is much more likely to make predictions as it was trained on a much smaller set of negative instances. Thus it is more likely to identify starts or ends that the L1 classifier missed.

4 Experiments

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

Evaluation method. A truly comprehensive comparison would compare each algorithm on the same dataset, using the same splits, and the exact same scoring method. Unfortunately, a conclusive comparison of the different IE algorithms is impossible using the published results. The other algorithms are evaluated using slightly different methodologies [7] or simply do not report results for every corpus.

There are several orthogonal issues regarding evaluation such as whether to give credit for partial matches and whether all occurrences of the field must be extracted. Our evaluation is conservative and so it is likely that our results are understated in comparison to competitors which have adopted a more liberal evaluation strategy.

Experimental setup. We evaluate our algorithm on three standard benchmark datasets, the seminar announcements (“SA”) dataset [5], the job postings (“Jobs”) dataset [2], and the Reuters corporate acquisitions (“Reuters”) dataset [5], using 31 fields (see Fig 3). SA consists of 485 seminar announcements annotated for 4 fields detailing upcoming seminars. Jobs consists of 300 newsgroup messages detailing jobs available in the Austin area. The dataset has been annotated for 17 fields. Reuters consists of 600 Reuters articles describing corporate acquisitions. This dataset has been annotated for 10 fields. It is a more difficult task than the other two because some of the fields are related. For example, there are separate annotations for the name of a company and abbreviated versions of the company name.

We used a 50:50 split of the dataset repeated 10 times. All experiments use a window of length 3 tokens, and L2 lookahead/lookback of 10 tokens. On the SA dataset, 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.

We compare our system against BWI, RAPIER, LP^2 and SNOW-IE using the available published results for each system. Many of the systems have not published results for all the fields.

Experimental results. Fig. 3 compares the performance of L1 and L2. The L1 results are measured by passing the predictions of the L1 classifiers directly to the Tag-Matcher, bypassing the L2 classifiers (see Fig. 2). All fields are shown together. The first 4 are from SA, the next 17 from Jobs and the last 10 from Reuteurs. On every field L1 has equal or higher precision than L2. On every field, L2 has higher recall than L1.On several fields, especially those with lower performance, the increase in recall is large. There are only three fields where L1 has higher f-measure than L2. Thus L2 causes precision to drop and recall to rise. But in most cases the increase in recall is greater than the drop in precision giving a corresponding increase in f-measure.

Fig. 4 compares the precision, recall and f-measure of ELIE$_{L2}$ with BWI,RAPIER, LP^2 and SNOW-IE. For each graph the horizontal axis shows the performance of ELIE$_{L2}$ while the vertical axis shows the performance of the other IE systems. Points below the diagonal indicate that ELIE$_{L2}$ outperformed the particular IE system on a particular field. Points above the diagonal indicate that the competitor IE system outperformed ELIE$_{L2}$.

On recall and f-measure ELIE$_{L2}$ outperforms the other algorithms on most of the fields. Some of the other systems perform better than ELIE$_{L2}$ when precision is considered. However ELIE$_{L2}$ improved recall at the expense of precision and if the task requires high precision then ELIE$_{L1}$ can be used instead.

Learning algorithms and features To test how the learning algorithm and the different feature-sets contribute to performance, we evaluated ELIE with various learning algorithms and reduced sets of features on the SA dataset.

We compared SMO with several well-known learning algorithms: naive Bayes, Winnow [8] and Ripper [4]. For SA, naive Bayes performs quite poorly on all fields. Winnow performs well on the etime field, but poorly on the other fields. Ripper performs well on all fields and is competitive with SMO.

Experiments using different feature-sets showed that for the location, stime and etime fields most of the performance comes using the token features alone. For each of these fields adding the POS, GAZ or orthographic features only gives a small increase in performance.

For the speaker field, we get F1=65.0% using only the token features, compared to 88.5% using all features. Using either the POS or orthographic features in addition to the token features gives an 7% increase in performance, while using the token and GAZ features gives a 16% increase over using token features alone. This indicates that the use of a gazetteer significantly enhances performance on this field. This is unsurprising as the gazetteer contains a list of first and last names. However the addition of POS and orthographic also provide significant benefit. On most fields the gazetteer provides little benefit. However for most fields our gazetteer did not contain information relevant to that field. An appropriate gazetteer list could improve performance on several fields but in general this may involve significant additional effort.

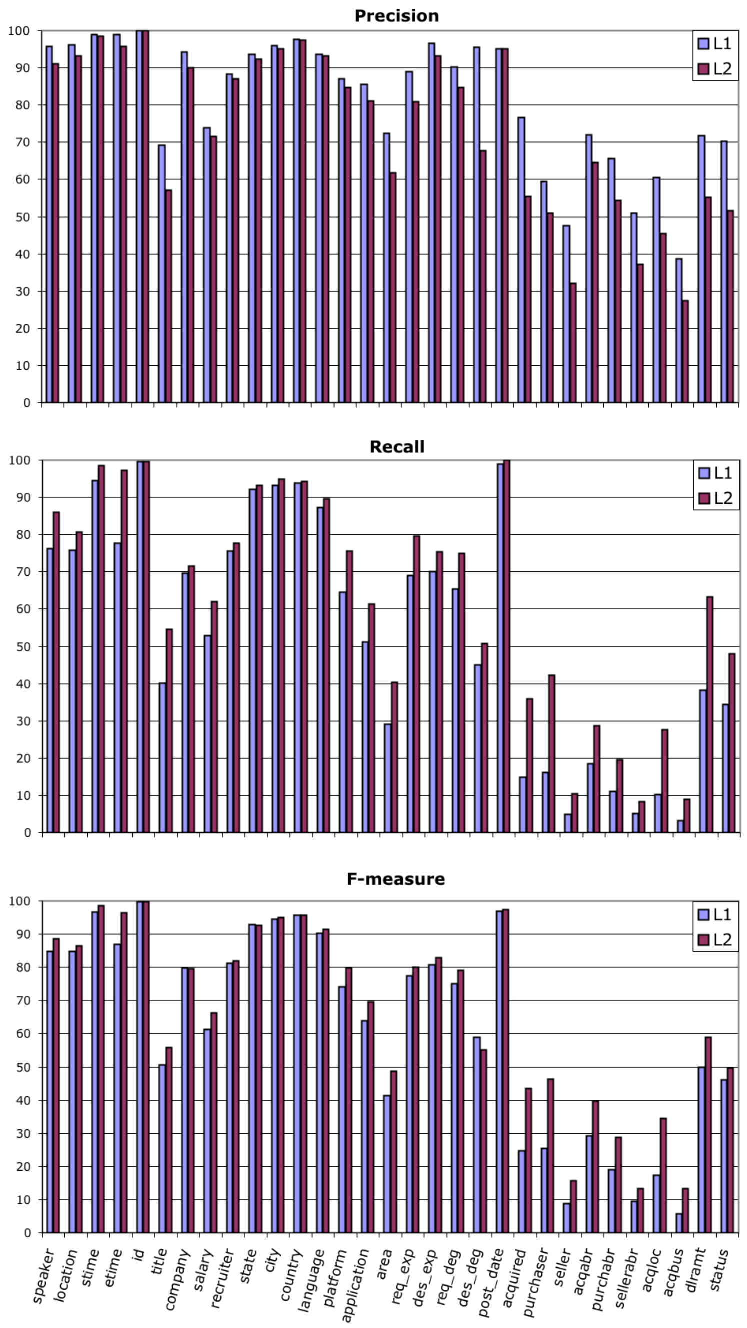

Fig. 3. Comparison of performance of L1 versus that of L2

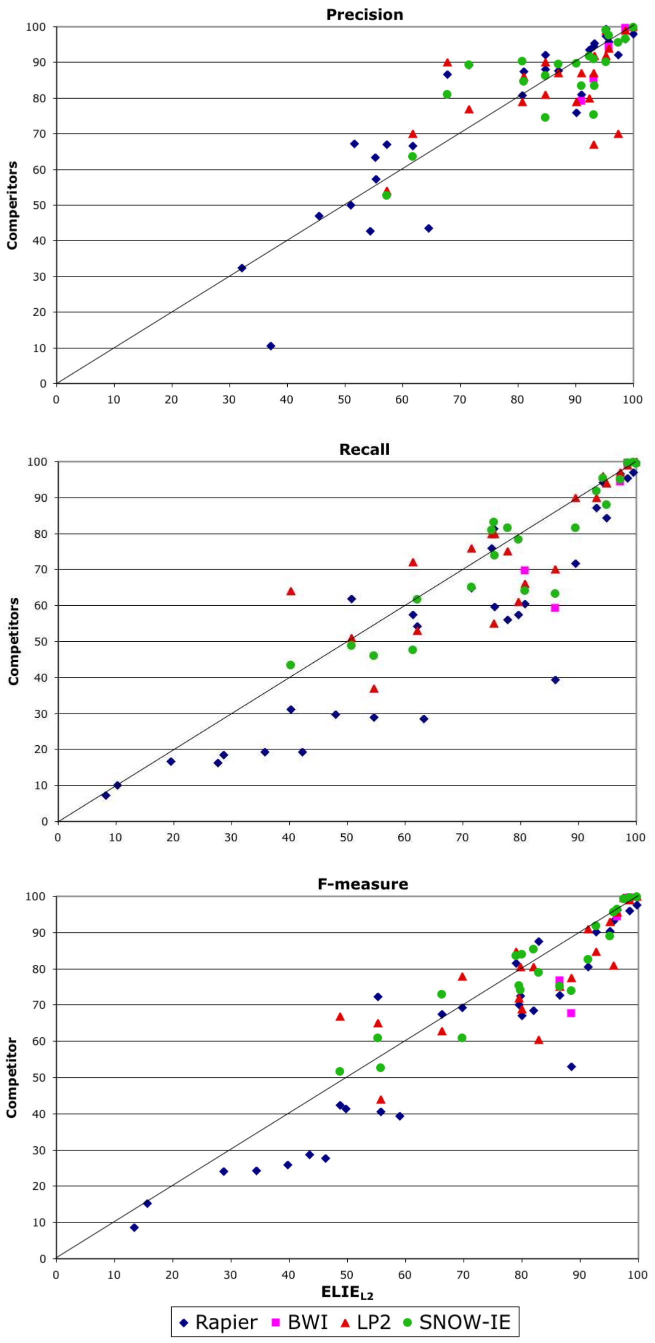

Fig. 4. Comparison of Elie$_{L2}$ with other IE systems

		L1			L2		
Dataset	Field	FP:FN	%FP$_{ptl}$	%FN$_{ptl}$	FP:FN	%FP$_{ptl}$	%FN$_{ptl}$
SA	speaker	0.17	22	62	1.05	17	8
SA	location	0.19	76	67	0.51	75	20
SA	stime	0.2	27	86	4.72	9	36
SA	etime	0.05	64	92	0.93	36	18
Jobs	id	0	0	100	0	0	100
Jobs	title	0.29	71	58	0.9	56	23
Jobs	company	0.14	9	10	0.27	14	2
Jobs	salary	0.4	76	68	0.66	68	43
Jobs	recruiter	0.41	22	22	0.52	21	11
Jobs	state	0.79	9	24	1.11	9	6
Jobs	city	0.56	1	28	0.95	1	1
Jobs	country	0.36	0	6	0.44	0	3
Jobs	language	0.25	41	45	0.52	30	10
Jobs	platform	0.27	43	43	0.54	37	10
Jobs	application	0.18	23	27	0.38	14	3
Jobs	area	0.15	34	25	0.41	25	6
Jobs	req_exp	0.28	8	41	0.92	6	9
Jobs	des_exp	0.09	100	10	0.23	54	12
Jobs	req_degree	0.21	0	34	0.53	2	1
Jobs	des_degree	0.04	0	10	0.51	5	0
Jobs	post_date	4.8	0	100	∞	0	0
Reuters	acquired	0.05	32	32	0.45	18	3
Reuters	purchaser	0.13	10	35	0.7	8	3
Reuters	seller	0.06	1	6	0.24	2	0
Reuters	acqabr	0.09	9	14	0.22	8	1
Reuters	purchabr	0.07	5	11	0.20	8	1
Reuters	sellerabr	0.05	1	4	0.15	2	0
Reuters	acqloc	0.07	16	27	0.46	16	3
Reuters	acqbus	0.05	29	14	0.26	25	4
Reuters	dlramt	0.24	27	53	1.39	15	14
Reuters	status	0.22	23	35	0.87	21	8

Fig. 5. ELIE error analysis.

ELIE error analysis. Table 5 shows details of the errors made by ELIE. For all fields in the three benchmark datasets we show the ratio of false positives to false negatives (FP:FN). It also shows the percentage of false positives that were partially correct and the percentage of false negatives that were partially predicted.

For a false positive to be partially correct means ELIE extracted a fragment, but that it was correct at only one end (either the start or end was not predicted exactly). These kinds of predictions are still useful in a practical setting and a less conservative method of evaluation might give some credit for these kinds of errors. On several fields, a large proportions of the errors are of this form.

For a false negative to be partially predicted means that for a fragment that we failed to extract, we predicted either the start or the end correctly, but may not have predicted the other. These are the kinds of errors that facilitate the improvement shown by L2 over L1. In general L2 gives a large reduction in these partial errors.

The ratio FP:FN shows that at L1, most of the errors are false negatives, while at L2 we generally see an increase in false positives and a reduction in false negatives.

Discussion and summary Our system outperformed those compared against on most fields in terms of recall or f-measure. If high precision is required then ELIE_{L1} can be used. We evaluated our system conservatively so its performance may be understated in relation to competitors.

The L2 learner consistently improves recall while keeping precision high. On more difficult fields 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.

It is likely that the accuracy of ELIE has two main sources. Firstly, since the L1 classifier alone often gives better performance than other IE algorithms, we conclude that the use of support vector machines as the learning algorithm gives rise to substantial improvement compared to the specialized learning algorithms used by most IE algorithms. Secondly the two-level classification that we have described can give significant increases in performance. It increases recall while maintaining good precision. In many cases, L2 improves ELIE's L1 performance substantially.

5 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 algorithm learns a set of classifiers for information extraction that are competitive with, and in many cases outperform, current IE algorithms based on specialized learning algorithms.

We also described multi-level 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 algorithm. Our algorithm outperformed current IE algorithms on three benchmark datasets. On several fields, especially those that are more difficult, it gave large improvements in performance.

There is 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 algorithm and/or documents facilitate this.

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.

Performance may be improved by changing how ELIE combines L1 and L2 predictions. Currently ELIE uses all the L1 and L2 predictions. However it might be feasible to use the L2 predictions to identify incorrect predictions from L1 and remove them. Finally, a more sophisticated tag-matcher could improve overall performance.

Acknowledgments. This research was funded by Science Foundation Ireland and the US Office of Naval Research. Thanks to Fabio Ciravegna for access to LP^2.

References

1. Eric Brill. Some advances in transformation-based parts of speech tagging. In *AAAI*, 1994.
2. Mary Elaine Califf and Raymond J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proc. 16th Nat. Conf. Artifical Intelligence*, 1999.
3. Fabio Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In *Proc. 17th Int. Joint Conf. Artificial Intelligence*, 2001.
4. William Cohen. Fast effective rule induction. In *ICML*, 1995.
5. Dayne Freitag. *Machine Learning for Information Extraction in Informal Domains*. PhD thesis, Carnegie Mellon University, 1998.
6. Dayne Freitag and Nicholas Kushmerick. Boosted wrapper induction. In *Proc. 17th Nat. Conf. Artificial Intelligence*, 2000.
7. Alberto Lavelli, Mary Elaine Califf, Fabio Ciravegna, Dayne Freitag, Claudio Giuliano, Nick Kushmerick, and Lorenza Romano. A critical survey of the methodology for IE evaluation. In *4th International Conference on Language Resources and Evaluation*, 2004.
8. Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.
9. Leonid Peshkin and Avi Pfeffer. Bayesian information extraction network. In *Proc.18th Int. Joint Conf. Artifical Intelligence*, 2003.
10. John C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.
11. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1993.
12. Dan Roth. Learning to resolve natural language ambiguities: A unified approach. In *National Conference on Artificial Intelligence*, 1998.
13. Dan Roth and Wen-Tau Yih. Relational learning via propositional algorithms: An information extraction case study. In *17th International Joint Conference on Artificial Intelligence*, 2001.
14. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.