Information Extraction from Online Automotive Classifieds

Matthew Rubens
Dept. of Computer Science
Stanford University
mrubens@stanford.edu

Puneet Agarwal
Dept. of Computer Science
Stanford University
agarwalp@stanford.edu

ABSTRACT

In this paper, we apply a combination of known natural language processing and machine learning classification algorithms to a real-world information extraction task, namely extracting attributes from online automotive classifieds. This type of information could be used to facilitate a structured search over the unstructured data.

1. INTRODUCTION

1.1 Motivation

The motivation behind this paper was the difficulty that one of the authors faced when trying to find a used car on the Internet. Many of the most popular websites selling used automobiles, such as the craigslist.com bulletin board popular in the San Francisco area, present a very basic keyword-based search interface. Many sellers capitalize on this keyword search by adding popular keywords such as "civic", "Toyota", etc. to their ads even if these keywords are completely irrelevant to the actual for sale content. Additionally, the basic search interface makes it difficult to search for desired attributes of a vehicle without listing myriad alternative representations. In general, it would useful for a buyer to have the ability to do a structured attribute-based search over the corpus.

1.2 Problem Definition

This task can be formulated as an information extraction problem, defined as the problem of populating a template with segments extracted from free text. Information extraction problems can be characterized as single-slot or multiple-slot extraction problems depending on the number of possible correct answers for a given attribute from a single document.

Since the end goal of our project is to facilitate an improved search over the data, we chose to extract what we considered to be five of the most frequently used attributes in the domain: year of manufacture, car make, car model, mileage on car, and asking price. It is possible for one post to have more than one correct value for each attribute: therefore our problem is a multi-slot one.

1.3 Approach

Broadly speaking, there are two general approaches taken to the problem of information extraction. The first approach is a rule-based one, relying heavily on domain experts to accurately express the patterns and constraints in the domain. The second approach is a learning-based approach where statistical models are applied to a moderate number of training instances to learn the patterns and constraints without the help of domain experts. Both approaches have advantages and disadvantages, and the choice of which to use depends on the resources available (i.e. time, domain knowledge, tagged data).

For this project, it seemed that a compromise between the two approaches was most appropriate. We were not able to find any relevant pre-tagged data, but we did have a practically infinite repository of unlabeled data to work with. Also, we have a layman's knowledge of automotive classifieds, but neither of us have extensive experience with the domain.

We used a freely available named entity extractor to generate a list of candidates for each attribute based on linguistic models, and then passed these attributes to a classifier which applied domain-specific and document-wide features to label the candidates as relevant or irrelevant. Thus we have statistical models of a named-entity extractor and a classifier augmented with domain knowledge in the form of dictionaries and hand-crafted domain-specific classifier features.

1.4 Related Work

Information extraction as a classification problem has been successfully attempted previously [1] [2]. Kushmerick et. al [1] effectively performed classification-based single-slot information extraction on business cards and change-of-address emails using a combination of a Hidden Markov Model and a Naïve Bayes model. Collins [2] used global features as input to a classifier to re-rank the output of a named entity extractor on a general question-answering problem. Our approach is more comparable to the latter.

2. DATASET

2.1 Structure

Our dataset consisted of postings from the "craigslist | cars & trucks in san francisco bay area" section of the popular craiglist online community. We downloaded the posts preserving the HTML format.

The postings turned out to be fairly unstructured, with the only required elements being a posting title, a posting description, and the seller's email address. There was also an optional price element that most posters used.

2.2 Challenges

The domain and attributes that we chose presented many challenges in determining which candidate values were relevant for a given attribute. To give a sense of the ambiguities faced, we have included a sampling of difficult training examples that we encountered.

In this example, our system must determine that although there are two years and two models mentioned in the post, only the 1967 Camaro is actually for sale.

1967 Camaro Project Car Rebuilt 350, TH-350 , Cheap - $3500

Asking $3,500. Must sell to pay a loan and I want to get a 1989 mustang as well. My loan is due soon as to i got an extension but was still un-able to pay the loan.

Figure 1: Example of irrelevant years and models

In this post, two mileage figures of similar magnitude appear in close proximity, although only one refers to the actual mileage on the car.

1996 Nissan Sentra - Clean - Great Gas Mileage! - $3700

I'll take $3700 or best offer. This car is in great shape and comes with all the extras. It has 102K on it but obviously will go for another 100K. I use it as a commuter car and it gets between 35 - 38 MPG. It's a 5 speed manual.

Figure 2: Example of ambiguity in detecting mileage

Even though this poster has misspelled 'Civic' as 'Cicvic' in several places, our system must still recognize that Civic is a model. Furthermore, this post has many mileage candidates, of which only one is relevant.

2000 Honda Cicvic HX - $8500

2000 Honda Cicvic HX2000 Honda Civic HX Coupe

…

75K miles, all freeway. I only use this car as a commuter to drive to from North bay to Oakland for work during the weekdays. My car gets about 500 miles to the tank. It has EXCELLENT fuel economy. That's almost 50 mile per gallon! Excellent Condition inside out!

Figure 3: Example containing a spelling error and mileage ambiguities

This post contains multiple tokens that are easily identified as prices, but only one refers to the actual asking price for the car. Our system must mark the other prices as irrelevant.

1999.5 Jetta GLS VR6 43k miles Quick sale! - $10500

I have for sale a 1999 1/2 (new body style) Jetta GLS VR6.

…

The on line blue book showed the trade in value as about $10,500, and the private party sale price of $11,900. If someone comes quick (as I am supposed to pick up the new car in the next day or so), I would be happy to sell it for the $10,500 price. E-mail or call me if interested.

Figure 4: Example of price ambiguities and year normalization

3. SYSTEM DESCRIPTION

3.1 Schematic

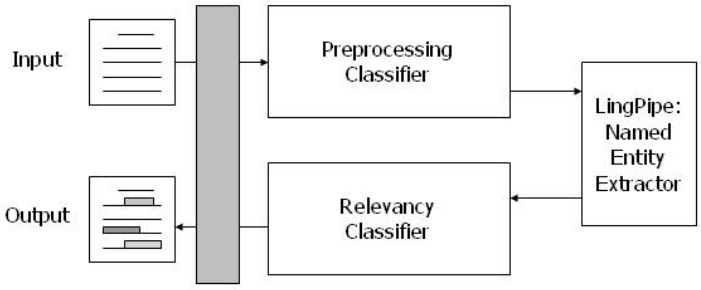

Figure 5: Illustration of the components of the overall system

Input documents are fed to a pre-processing classifier which prunes out the 'want to buy' posts and the 'selling parts' posts. The remaining set is input to the LingPipe module which augments the text with named-entity information and passes the output to a relevancy classifier. This classifier determines which of the entities are relevant and return the fully marked up document as the output of the overall system.

3.2 Training

We began by hand-annotating 300 postings with their named entity tags using a modified version of a Java-based document annotation system (part of the JavaNLP intitiative at Stanford Univeristy) generously provided to us by Chris Manning. To facilitate training both a named entity extractor and a classifier we marked both correct and incorrect instances of all attributes in the training data.

3.3 Named-Entity Extraction

We used a freely available linguistic analysis toolkit called 'LingPipe' provided by Aliasi-I Inc. for the named entity Extraction task. It uses a generative model decomposed into a tag and lexical model and uses previous word/tag pairs, word trigram and tag bigram history to predict the next word/tag pair. To adapt it to the problem on hand, we modified the tokenizer to correctly tokenize prices (beginning with a '$' for example) and trained the model on our dataset. The input to the extractor was stripped of all HTML information to avoid misleading it into building an incorrect language model.

3.4 Data Normalization

The data that we saw contained many alternative representations (including spelling mistakes and abbreviations)

for the same concept. We believe that this problem is compounded by the fact that opposed to newspaper classifieds, which impose a cost constraint on the ad, online bulletin boards such as craigslist are free and therefore do not provide much incentive for posters to spell-check their posts, as they can always post again for free if they so desire. An additional constraint in the print world that does not apply to this medium is that newspapers impose a length constraint on the advertisement. However, a craigslist poster can make their post arbitrarily long (and is in fact encouraged to do so by the keyword search) and therefore can include much more irrelevant information.

Since many of features depend on term frequency and the actual value of the numeric attributes, we needed some method by which to deal with these alternative representations. Our approach to this problem was twofold.

For text attributes (Make and Model), we used approximate string matching algorithms with different similarity metrics such as Levenstein edit distance, Jaro-Winkler, and Monge-Elkan [5]. This took care of misspellings such as "Civic" vs. "Cicvic" (Figure 3) and subtle differences such as "Corolla CE" vs. "Corolla LE". We used the SecondString library freely available at http://secondstring.sourceforge.net/ for this purpose. The Jaro-Winkler metric seems to make a tradeoff between accuracy and speed compared to the Monge – Elkan metric which is probably more effective since it is based on an affine distance gap function.

For numeric attributes, we had to perform class-specific normalization to map the varying representations of each class to a common numeric representation. To give an example of the problem at hand, consider these tokens:

Year: "98, '98, 98', 1998, "98"

Price: $5,000, $5,000.00, $5000, $5.3K, 5000 dollars, 5000$

Mileage: 11,000 miles, 11K, 11kmiles, 11 kmiles, 11xxx mi, 11KM

3.5 Feature Selection

The output of the Named-Entity extractor is a set of candidate segments of text marked as Make, Model, Mileage etc. and the following problem now is to choose the most relevant one among these which corresponded to the post. Our approach here was to train a classifier over the tagged training set with features both at the document as well as the contextual level to allow it to distinguish the relevant ones from the irrelevant ones.

3.5.1 Feature Descriptions

The features that we chose can be divided into 3 broad categories (for description see Table 1):

(i) Document-level (global) features

Description: These features intended to exploit the document structure and thus capture information provided by the overall document and the relative position of the tag in question within the document. These are general enough to apply to all attributes.

NearBeginning	How close the candidate is to the beginning of the document, scaled by the document's length.
TermFrequency	How many times the candidate occurs in the document
NumberPreceding	How many other candidates with the same tag precede the candidate in question in the document
NumberPreceding_diff	How many other candidates with the same tag but different normalized values from the candidate in question precede it in the document
Order	NumberPreceding scaled by the total number of candidates for the given attribute
NumberSucceeding	How many other candidates with the same tag succeed the candidate in question in the document
NumberSucceeding_diff	How many other candidates with the same tag but different normalized values from the candidate in question succeed it in the document
SameAsFirst	Whether this candidate has the same normalized value as the first candidate for this attribute to appear in the document
TagMatch_[term]_ [leftContext]_[rightContext]	Whether a token with the given tag term occurs within the specified context of the candidate in question
WordInContext_[term]_ [leftContext]_[rightContext]	Whether the given term occurs within the specified context of the candidate in question
MakeMatchScore	The Jaro-Winkler similarity between the Make candidate and the closest entry in the precompiled dictionary of makes
ModelMatchScore	The Jaro-Winkler similarity between the Model candidate and the closest entry in the precompiled dictionary of models
MatchingMakePreceding	Whether the Model candidate is preceded by a matching Make
ScaledPrice	Feature corresponding to the magnitude of the price
ScaledMileage	Feature corresponding to the magnitude of the mileage

Table 1: Description of Features used for classification

Examples:

- NearBeginning

  Upon examination, it seemed that most posts gave more structured, predictable, and relevant information near the beginning of the document.

- NumberPreceding_diff

  Since most documents only have one unique value for each attribute, the number of distinct attributes tagged by the named entity extractor provides an estimate of the ambiguity of the document.

- TermFrequency

  Terms that occur more often in the document tend to be more relevant than terms that occur infrequently.

(ii) Hand-crafted context-level features

Description: We used our (limited) domain knowledge to come up with contextual features that would allow us to gauge the relevancy of the tags.

Examples:

- TagMatch_Year_2_0, TagMatch_Make_1_0

  The pattern [Year] [Make] [Model] is a good indicator that this sequence of tokens are referring to a specific vehicle, likely one which is the subject of the post.

- WordInContext_-_1_0

  We use the domain knowledge that when a user chooses to fill in the price field for the post, this price always occurs in the post title preceded by a dash. Therefore, it is very likely that prices preceded by a dash are relevant.

- MakeModelMatch

  Since a Make and a Model are not truly independent of each other, we tried to include a feature which captures this notion. We do an approximate match using the Jaro-Winkler metric to compute the closest model in the dictionary. We then look in the preceding context for a Make tag where the given Model in question is manufactured by this Make. Note that we only look as far back as the previous Model tag.

- MakeMatchScore for Make, ModelMatchScore for Model

  We used a dictionary of common Makes and Models from Yahoo! Autos and computed the closest score based on the Jaro-Winkler metric. This allows us to weed out incorrect phrases marked by the NE extractor as a candidate Make or Model.

- WordInContext_keyword_20_0 for Make

  Because of the simple textual keyword search currently in use on craiglist, users have taken to providing a list of popular search terms at the bottom of the document in hopes of attracting more traffic to their posting. Therefore, this feature indicates that the make is less likely to be relevant.

(iii) Automatically discovered features

Description: We generated the vocabulary from the training instances and created features for all of them within a given context to both sides. We then used Information Gain to do feature selection. The learned relevant terms often captured domain characteristics that we were previously unaware of.

Examples:

- WordInContext_registered_5_5 for Year

  It turns out that a lot of posts give information about the validity period of the registration. The years occurring in registration dates comprised a large portion of the negative examples.

- WordInContext_[rebuilt|timing|warranty]_5_5 for Mileage

  A large of number of negative Mileage examples came from phrases like “40K miles on rebuilt engine”, “timing belt changed at 100K”, “warranty for 5 yrs/50K miles”.

3.5.2 Feature Evaluation

We ranked the different features using an Information Gain metric over the training data to evaluate their effectiveness and prune unnecessary features. The results are presented below:

0.072	NumberPreceding_diff
0.0438	NearBeginning
0.0363	TermFrequency
0.0339	NumberPreceding
0.0315	TagMatch_Year_1_0
0.0302	WordInContext_keyword_20_0
0.0225	NumberSucceeding_diff
0.0141	Order

Table 2: Information Gain for ‘Make’ features

0.1777	NumberPreceding_diff
0.14524	NearBeginning
0.10743	NumberPreceding
0.10428	TermFrequency
0.08295	WordInContext_keyword_20_0
0.08176	NumberSucceeding_diff
0.05932	Order
0.05098	TagMatch_Year_2_0
0.03692	NumberSucceeding
0.01946	TagMatch_Make_1_0
0.01733	MatchingMakePreceding
0.00566	TagMatch_Color_5_5

Table 3: Information Gain for ‘Model’ features

0.1407	SameAsFirst
0.1172	NumberPreceding_diff
0.0901	NearBeginning
0.0804	NumberPreceding
0.0573	TagMatch_Make_0_2
0.0542	Order
0.0497	TagMatch_Model_0_4
0.0389	TermFrequency
0.0256	WordInContext_registered_10_10
0.0256	WordInContext_until_10_10
0.025	WordInContext_in_10_10

Table 4: Information Gain for ‘Year’ features

0.1547	SameAsFirst
0.1349	NumberPreceding_diff
0.0703	WordInContext_book_10_3
0.0648	NumberPreceding
0.0565	WordInContext_-_1_0
0.0542	NearBeginning
0.0505	Order
0.037	TermFrequency
0.0297	ScaledPrice
0.0246	WordInContext_kbb_10_3
0.0222	NumberSucceeding_diff
0.0119	WordInContext_obo_0_4

Table 5: Information Gain for 'Price' features

0.0993	SameAsFirst
0.0649	NumberPreceding_diff
0.0557	ScaledMilage
0.0415	Order
0.0385	WordInContext_rebuilt_10_10
0.0377	NumberPreceding
0.0332	WordInContext_tank_10_10
0.0332	WordInContext_gets_10_10
0.0312	WordInContext_engine_10_10
0.0269	WordInContext_timing_10_10
0.0269	WordInContext_belt_10_10
0.0246	WordInContext_warranty_10_10

Table 6: Information Gain for 'Mileage' features

As is indicated by the above figures, the 'domain knowledge' introduced by our hand-crafted features seem to be quite important in terms of information content. The automatically 'learnt' features figure at the lower end of most tables above and are indicative of the fact that individual words in context are not all that informative to gauge the relevancy of a candidate name-entity pair.

3.6 Classification

A subtle issue here was the formulation of the classification problem i.e. should the classifier try and rank the candidate pairs on some kind of confidence measure or should it simply output a 'relevant' or 'irrelevant' decision on each one. We determined that a binary classification was probably more suited to the problem domain given the fact that a post could have zero, one or many relevant choices as opposed to exactly one relevant choice. For example, a post which said 'Selling my Mustang, wanna buy a Honda' actually has no 'relevant' Makes mentioned and one 'relevant' Model mentioned. However, it does have an 'irrelevant' Make which would still be ranked as the most relevant by a ranking classifier and could be misleading. On the other hand posters selling more than one car have multiple 'relevant' Makes and Models mentioned and its difficult to rank these in terms of relevance.

We tried out different classifiers like a linear SVM, an AdaBoost Decision Tree with 10 boosting iterations, a Naïve Bayes model and a Voted Perceptron model and report classification accuracy via 10-fold cross validation on the training set of 300 instances. We used a freely available library called 'Weka' from http://www.cs.waikato.ac.nz/ml/weka/ [3] for all our classification and feature selection tasks. The results are tabulated below in Table 7.

%	SVM	AdaBoost Decision Tree	Naïve Bayes	Voted Perceptron
Make	98.6	98.6	97.1	97.8
Model	98.3	98.5	96.6	98.1
Price	95.6	95.2	94.3	95.2
Year	98.7	98.0	97.9	98.7
Miles	91.2	89.8	90.8	91.5

Table 7: Comparison of different statistical classifiers

3.7 Rationale:

It was evident from looking at the posts that there was no real structure to the language used. Most posts consisted of fragmented phrases rather than fully-formed sentences. Hence clearly syntactic parsing based methods would not prove very useful in dealing with the given data. Instead a rather specialized kind of 'language' was used throughout a majority of the posts which was perhaps a characteristic of the popularity of the domain and hence a word-based language model which leveraged an n-gram model would probably turn out to be more useful. Thus an initial named entity extractor which could learn from the language commonly used in the examples and predict candidate sets for us to focus on seemed a likely choice. This however does introduce a requirement of high recall on the extractor since it 'introduces' candidate sets to the system.

Given that LingPipe already looked at the previous word trigram and tag bigram history to predict the next token/tag pair, it is instructive to look at the value of the hand-crafted features introduced. After all if the trigram and bigram history was enough for the problem, LingPipe could itself have achieved results as good as possible. This is certainly not the case since the NE extractor fails to capture information provided by the inter-dependence of attributes, document structure, and information captured by the presence/absence of certain entities within a larger context. Hence, features like MakeModelMatch which try to exploit the dependence of the 'Model' and the 'Make' of the car based on a pre-compiled dictionary are useful to recognize relevant 'Models'. Similarly the presence of multiple 'Make' entities within the document is an indication of the level of ambiguity possibly present and serves to adjust the confidence measures for tagged 'Models'. Also the fact that a sell post probably talks about the relevant product before talking about other irrelevant ones means that entities encountered earlier on in the document are probably more predictable and relevant. Features like these are not captured within the limited window history that LingPipe

maintains and is effectively utilized by our classifiers to gauge relevancy.

Finally, although our original input was in HTML, we converted to a text representation and stripped out the HTML tags. We did not see any major information value in the layout or presentation of the document which could have been present in the HTML. It also had to be removed to allow the named entity extractor (which had a documented weakness in dealing with HTML data) to effectively learn a prediction model.

4. EVALUATION

We manually tested all parts of our system on a test set of 100 new instances. To report performance, we use the following well-known metrics:

$$precision = \frac{tp}{tp + fp}$$

$$recall = \frac{tp}{tp + fn}$$

$$F - Measure = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

4.1 Preprocessing:

Since we were looking only at the ads that were selling automotives (cars/trucks), we used a classifier trained on a bag of words model to prune out the 'want to buy' posts and posts selling parts instead of autos. We ran feature selection using Information Gain over the vocabulary built up over the training set to come up with a set of 'useful' features. We ran an SVM with a higher (3-degree) polynomial kernel, an AdaBoost Decision Tree with 10 boosting iterations and a Naïve Bayes classifier on it with the following cross-validation (10-fold) accuracy:

Accuracy (%)	SVM	AdaBoost Decision Tree	Naïve Bayes
Cross-Validation (10-fold)	97.00	96.33	96.67

Table 8: Choice of classifier for Buy/Sell

Similarly, we repeated the above to prune out the posters selling parts instead of autos. Results are reported below for 10-fold cross-validation on the training set of 300 instances.

Accuracy (%)	SVM	AdaBoost Decision Tree	Naïve Bayes
Cross-Validation (10-fold)	98.67	98.67	88.33

Table 9: Choice of classifier for Parts/Auto

Based on the cross-validation accuracy, we chose the SVM with the higher (3-degree) kernel for our final system. We report classification accuracy on the test set of 100 instances below:

Accuracy (%)	Buy/Sell	Parts/Auto
SVM	98.00	100.00

Table 10: Classifier accuracy on Test Set

The performance on the test set reinforces the fact that classifying a buy vs a sell post and an auto vs a parts post is a relatively simple problem which is handled well by the sophisticated classifiers.

4.2 LingPipe: NE Extractor

We evaluated the NE extractor on the test set and report the following precision and recall:

LingPipe	Precision (%)	Recall (%)	F-Measure
Make	98.48	98.98	98.73
Model	95.81	92.33	94.04
Year	98.55	98.08	98.31
Price	98.41	96.88	97.64
Mileage	89.47	93.41	91.40

Table 11: Evaluation of LingPipe over test data

As the above table indicates, LingPipe seems to perform quite impressively over the test set. This is in spite of a relatively small and inconsistent hand-tagged training set of 300 examples. It should be noted though that we care more for the recall than precision in case of the NE extractor since the purpose of the NE extractor is to present a set of possible candidates to the classifier. The recall can be improved with a larger training set. This would help the overall F-measure too.

4.3 Classifier:

We note that there are fundamentally two types of errors that the NE extractor makes (in terms of its precision):

(i) Tokens that it tags which are true members of the entity class but are irrelevant in terms of the post (e.g. 'blue book price of $7K but am selling it at $6,500' - $7k is a Price entity but is not the price corresponding to the post)

(ii) Tokens that it tags which are not true members of the entity class itself (e.g. 'I am selling a 1991 Corolla' – where Corolla is incorrectly tagged as a 'Make' instead of a 'Model')

Both the above types of errors need to be classified as 'irrelevant' by the classifier to ensure accuracy of the overall system.

We tested a variety of classifiers which use different paradigms over the test set of 100 new hand-annotated instances and report the following figures:

SVM	Make	Model	Price	Year	Miles
Precision	94.4	89.5	86.5	97.6	84.2
Recall	100	100	99.1	99.5	98.7
F-measure	97.1	94.5	92.4	98.5	90.9

Table 12: Linear SVM classification performance

AdaBoost Decision Tree	Make	Model	Price	Year	Miles
Precision	95.5	90.8	88.9	96.6	86.3
Recall	100	100	98.1	98.9	96.0
F-measure	97.7	95.2	93.3	97.7	90.9

Table 13: Decision Tree with 10 boosting iterations classification performance

Naïve Bayes	Make	Model	Price	Year	Miles
Precision	96.0	91.2	88.1	97.1	88.4
Recall	99.5	97.6	88.0	98.4	96.0
F-measure	97.7	94.3	88.0	97.7	92.0

Table 14: Naïve Bayes classification performance

Voted Perceptron	Make	Model	Price	Year	Miles
Precision	94.4	90.0	86.5	96.1	88.4
Recall	100	100	98.1	98.9	100
F-measure	97.1	94.7	91.9	97.5	93.8

Table 15: Voted Perceptron classification performance

As the above tables indicate, almost all classifiers seem to perform comparably on the test set. The recall in most cases is better than the precision reported which is favorable given that our end goal is to improve search over the data. Hence, we would ideally want an overall recall of 100% to report all positive instances for a given query even if at the cost of some precision (similar to the spam classification problem).

With the above in mind, we considered a baseline system which would guarantee a recall of 100% by classifying everything as 'relevant'. We compare the performance in terms of the F-measure of the Adaboost Decision Tree (with 10 boosting iterations) classifier to this baseline system in the chart below:

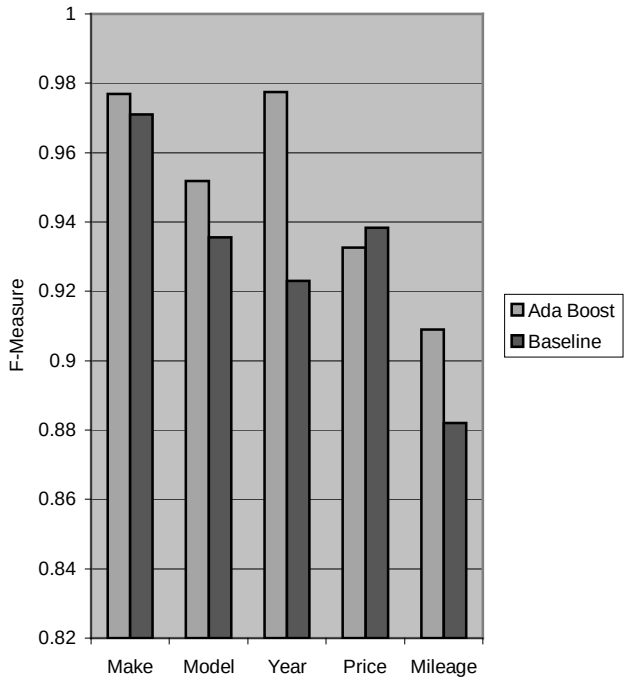

Figure 6: Comparison of AdaBoost to Baseline

The AdaBoost decision tree seems to outperform the baseline system on almost all attributes (except Price). The relatively low number of negative examples in the training data makes it more difficult for the classifier to be able to recognize them accurately. We believe that with a larger training set the performance of the classifiers could be increased significantly.

Finally, in terms of the overall system, it is instructive to note that given the sequential nature of the system, it is important to achieve high recall for the named entity extractor. Its performance with training on a small set of 300 training examples is encouraging and seems to suggest that a larger training corpus could be effectively employed to obtain better results.

5. FUTURE WORK

In this project we only attempted the problem of extracting information. This is a big step towards the goal of attribute-based search. However, a lot of possibilities remain in extending the work to better deal with normalization across domains and facilitating search over a wider range of attributes. In particular there are more difficult attributes like Color which do not have a well-defined domain and are used in highly ambiguous senses. This also hints at the possibility of including more knowledge in the system via larger dictionaries to effectively deal with a wider range of unseen data. Finally, a more immediate extension to allow it to deal with posts which are 'looking to buy' instead of sell and mention ranges instead of exact attributes is certainly possible.

6. CONCLUSION

In this paper we describe our system which successfully extracts information from online automotive classifieds accurately. In the process, we demonstrate how freely available open-source toolkits can be effectively used independently of each other to achieve high performance on a real-world problem.

7. ACKNOWLEDGMENTS

We would like to gratefully acknowledge the help we received from Prof. Manning in terms of the Annotator and insightful comments and feedback on our approach and Rajat Raina for useful tips along the way. We would also like to thank the providers of Weka (University of Waikato, New Zealand), Lingpipe (Alias-I Inc.) and SecondString (Carnegie Mellon University) without which this effort could not have been realized. Finally, thanks to craigslist.com for providing us with the data.

8. REFERENCES

[1] Kushmerik N. et al., *Information Extraction by Text Classification*

[2] Collins M., *Ranking Algorithms for Named-Entity extraction: Boosting and the Voted Perceptron*

[3] Ian H. Witten and Eibe Frank, *Data Mining: Practical machine learning tools with Java implementations*, Morgan Kaufmann, San Francisco, 2000.

[4] Cole, R., and Eklund, P. *Browsing Semi-structured Web texts using Formal Concept Analysis*.

[5] Cohen, W., Ravikumar, P., and Fienberg, S. *A Comparison of String Distance Metrics for Name-Matching Tasks.*

[6] LingPipe: http://www.aliasi.com/lingpipe/