Automatic Rule Generation and Generalization for an Information Extraction System Using MAXIM

J. McConnell
Swarthmore College
500 College Ave.
Swarthmore, PA 19081
mmcconn1@swarthmore.edu

Anteneh Tesfaye
Swarthmore College
500 College Age.
Swarthmore, PA 19081
atesfay1@swarthmore.edu

Abstract

Many recent information extraction (IE) systems have ignored the tedious and time-consuming nature of the preparation involved in using them. The abundance of graduate students has eased the pain of providing annotated corpora, pre-filled answer templates, and manual examination of automatically-generated rules and final answers. In this paper, we present a new system comprised of previously published solutions to different aspects of IE in an effort to automate as much of the task as possible while achieving competitive results.

1 Introduction

1.1 Background Information

The recent availability of large quantities of text in electronic format on the World Wide Web, has greatly increased the importance of intelligently extracting desired information from such text. The task of information extraction is most easily described as that of filling a database with information found in structured, semi-structured, or free text. Structured text can be thought of as text in a pre-defined format, like in a printed spreadsheet. Semi-structured text follows general guidelines, but is not as predictable as structured text. It is often ungrammatical with a lot of fragmented sentences. An example of such text might be telegraphic, military communications, or birthday invitations. Free text is just normal prose, as found in a news article or a work of fiction, for example.

DARPA recognized the significance of this growing field in the late 1980's when they funded the first Message Understanding Conference (MUC-1). The MUC has been held semi-annually since, and has highlighted developments in IE research. Since the early knowledge-engineered (KE) systems developed for MUC-1, the field has seen a trend towards automation to circumvent the bottleneck associated with KE.

This trend is fueled by the difficulties inherent in all KE tasks. Extensive, yet required, human involvement makes them costly to develop, test, and apply to different problem domains both in time and money. These systems often require annotated corpora, pre-filled answer templates, human designed rules, or, if the systems automate the rule-making process, the manual examination of such rules to weed out the poor ones. These requirements are simply unacceptable for many potential applications, and we believe they are unnecessary.

1.2 Relevant Work

Since MUC-1, researchers have been searching for an effective, autonomous IE system. Showcased at MUC-4, Riloff's AutoSlog (1993) program automatically generated a dictionary of concepts which were later used to extract information from text similar in category to those with which it was trained. This system proved capable of achieving 98% of the performance of a hand-crafted dictionary developed by graduate students. The students' dictionary took 1500 person-hours to build, while the AutoSlog dictionary only required 5 person-hours in order to hand-filter automatically-generated rules.

Despite the obvious benefits of AutoSlog, it was still not practical for real-world use. As input, AutoSlog required either a set of answer keys or a semantically-annotated corpus. This was used to provide AutoSlog with examples of information to be extracted. Consequently, AutoSlog does not port well to different domains since it takes many person-hours to either fill in a set of answer keys or annotate a corpus.

To address these concerns, Riloff developed AutoSlog-TS (1996). This improvement on AutoSlog automatically generated extraction rules given completely unannotated text. As input, it requires two texts, one relevant to the problem domain, and one completely irrelevant. It works by generating an extraction pattern for every noun phrase

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 84–91
Computer Science Department, Swarthmore College

in a given text. It then compares the extraction patterns found in the relevant text to those found in the irrelevant text. Those patterns that show up frequently in the former but not at all in the latter are presumed to be pertinent to the problem domain. This system is hindered however by the need for manual examination of the resulting rules, normally about 2,000, in order to discard poor choices.

Another system developed to automate the rule generation process is Rapier (Califf and Mooney, 1997). Rapier takes as input sets of training text paired with a corresponding answer key. It then uses this information, combined with the output of a POS tagger and semantic class tagger, each given the training text, to create specific extraction rules that extract the correct answers. Up to this point, Rapier is quite similar in execution to AutoSlog; however, the system then goes on to generalize these specific rules in order to make the resulting dictionary more robust. That robustness is the strength and the point of the Rapier system.

2 The MAXIM System

2.1 Basis for MAXIM

The Maximally Automated eXtractor of InforMation (MAXIM) system was developed out of the need for an IE method that required next to no human intervention or preparation but still received satisfactory results. AutoSlog-TS's necessity for manual examination of rules as well as its lack of robustness given the specificity of its resultant rules leaves something to be desired. Rapier's need for answer keys means many person-hours are required before training on any particular problem domain. The respective strengths and weaknesses of these systems complement each other well. As a result, we propose a joint system built with the better halves of both. This system consists of an implementation of the rule generation phase of AutoSlog-TS and a rule generalization process inspired by Rapier's rule representation and learning algorithm.

MAXIM takes as input pre-classified text from both inside and outside the problem domain.[1] It will feed this text to the rule-generation phase of AutoSlog-TS which has been modified to represent rules in a format similar to that of Rapier's (see Subsection 2.2). To minimize the time spent on the manual examination of the many resultant rules, we used a clustering algorithm in order to group similar rules. We were then required to examine only a fraction of the rules outputted by AutoSlog-TS.

[1] By "classified text", we mean a text that is marked relevant or irrelevant. We assume that finding qualified texts should not be difficult or time-consuming given that relevant text is required for training any system and irrelevant text is readily available online.

Pre-filler:	Filler:	Post-filler:
1) tag: {nn,nnp}	1) word: undisclosed	1) sem: price
2) list: length 2	tag: jj	

Figure 1: Sample Rule Learned by Rapier

2.2 Rule Representation

2.2.1 Rapier

Rapier's rules consist of three parts, a pre-filler pattern, filler pattern, and post-filler pattern. Each pattern consists of any number of constraints. Constraints can be defined to match an exact word, any word with a given POS tag, or any word that matches a given semantic class as defined by WordNet (Miller et al., 1993). Within constraints, expressions can be disjunctive if written as $\{constraint_1, constraint_2, \ldots\}$ where the constraints musts all be either exact words, POS tags, or semantic class tags. For example, to specify that a pattern should match with either an adjective or an adverb, the constraint would be {JJ, ADV}.

The appeal of Rapier's rule representation is its ability to express a general idea as opposed to relying on specific word choice. For example, a rule generated by Rapier for extracting the transaction amount from a newswire regarding a corporate acquisition is shown in Figure 1. The value being extracted is in the filler slot and its pre-filler pattern is a list of at most two words whose POS tag is either noun or proper noun. The post-filler pattern requires a WordNet semantic category "price".

2.2.2 MAXIM

Rapier's slot constraints form the underlying idea of the rule representation that we have adopted for MAXIM. Due to the inconsistencies between the methods used by AutoSlog-TS and Rapier to generate extraction patterns, we had to use the pre- and post-filler slots as containers for extracted values, which contrasts with Rapier's use of the filler slot for this purpose. This simple but crucial alteration in slot design meant that we could not use Rapier's rule generalization algorithm without modification. Also, the fact that this algorithm was highly dependent on the answer key provided to Rapier reinforced our decision to abandon this specific generalization algorithm entirely.

Our implementation of AutoSlog-TS returns the extraction patterns in the form of <noun phrase><verb phrase><noun phrase> which aligns nicely with the pre-filler, filler, and post-filler slot arrangement of the rule generalization phase. We set up three constraints for the pre-filler and post-filler slots and one constraint for the filler slot. The pre and post filler constraints consist of the maximum number of words in the noun

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 84–91
Computer Science Department, Swarthmore College

phrase, *max-len*, a 39-dimensional vector that serves as a histogram bin [2] for the POS tags of the words in the noun phrase, *POS-classification-vector*, and a required POS tag, *required-POS*, that is determined from the noun phrase's *POS-classification-vector*. The 39-dimensional vector was composed of 36 Penn Treebank part of speech tags and three tags that were added later to cover punctuation marks. The constraint associated with the filler slot is just a set of words in the verb phrase, *filler-set*. Like Rapier, MAXIM avoids this problem by generalizing rules using the slot constraints.

2.3 Problem Domain

Our problem domain is articles reporting the results of soccer games. It was chosen based on a mutual interest and the wide availability of such stories. In order to diversify our training corpus as much as possible while staying in the domain, we have collected stories from different countries and authors, resulting in texts with different writing styles. All articles are in English. Our sources include, the FIFA coverage of the World Cup 2002, the English Premiere League's archives of the past seven years, as well as the Major League Soccer's archives over the past six years. All sources were downloaded from the World Wide Web and have been stripped of all HTML tags.

Having considered various choices for the irrelevant text to be input into AutoSlog-TS, we decided that it would be beneficial to try different sources in order to compare and contrast how the choice of irrelevant text affects results. We have picked the Wall Street Journal corpus for its journalistic style and its specific and consistent subject matter. Conversely, we have chosen the Brown corpus for its broad range of writing style and diverse subject matter.

2.4 Building the Relevant Corpus

Once all the HTML tags had been stripped from our compilation of soccer stories, we performed two tasks in order to convert our relevant data to a corpus that was compatible with both the AutoSlog-TS and Rapier systems. The first task was to remove text that was not in a sentence format such as headers and game statistics. The second task was to put the soccer stories in one-sentence-per-line format. As a result, we implemented a highly customized sentence boundary disambiguator which included some common proper nouns from the problem domain. The fnTBL (Ngai and Florian, 2001) POS tagger and text chunker was then run on the formatted text, which included about 800,000 tokens.

[2]It is not technically a histogram, because the POS counts for the phrase are weighted. More common tags, like NN, for example, are weighted less than tags like CD, which carry more information.

PATTERN	**EXAMPLE**
<**subj**> passive-verb	<team> was defeated
<**subj**> active-verb	<player> scored
<**subj**> verb infin.	<team> attempted totie
<**subj**> aux noun	<player> was team
passive-verb <**dobj**>	kicked <player>
active-verb <**dobj**>	beat <team>
infin. <**dobj**>	to card <player>
verb infin. <**dobj**>	tried to foul <player>
gerund <**dobj**>	coaching <team>
noun aux <**dobj**>	coachName is <coach>
noun prep <**np**>	goal against <team>
active-verb prep <**np**>	beat by <goals>
passive-verb prep <**np**>	was injured at <time>

Figure 2: AutoSlog Heuristics

2.5 Implementing AutoSlog-TS

Though Riloff generously offered her AutoSlog-TS implementation for our research, we obtained the code too late to make use of it. Also, since modifications were necessary to the rule representation, time to become familiar with the code would be called for. For these reasons, we decided to implement what we needed of AutoSlog-TS ourselves. Due to the extensive need for using regular expressions and the limited time allotted for development, we decided to implement AutoSlog-TS with Perl. AutoSlog-TS generates extraction patterns for a given text in two stages.

2.5.1 Stage 1

In the first stage, AutoSlog-TS identifies the noun phrases using a sentence analyzer.[3] For each noun phrase, it uses 15 heuristic rules to generate extraction patterns that will be used in the second stage. These heuristic are shown in Figure 2.

When the first stage is run on the corpus, a huge dictionary of extraction patterns is created. This list of extraction patterns is capable of extracting every noun phrase in the corpus. AutoSlog-TS allows multiple rules to be activated if there is more than one match. This results in the generation of multiple extraction patterns for a single noun phrase. For example, running our implementation of AutoSlog-TS on a test set of the WSJ, the sentence "... have to secure additional information and reports ..." produced two patterns: **have to secure** <**dobj**> and **to secure** <**dobj**> in response to the two of the rules in

[3]We used the pre-trained englishTextChunker that comes as part of fnTBL.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 84–91
Computer Science Department, Swarthmore College

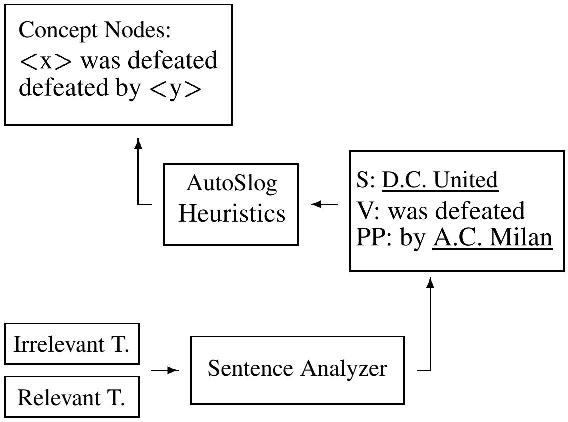


Figure 3: AutoSlog-TS Stage 1 flowchart

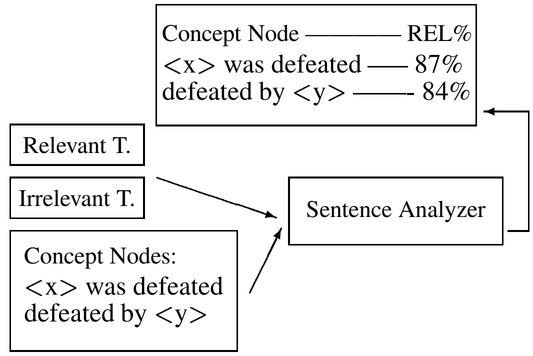


Figure 4: AutoSlog-TS Stage 2 flowchart

Figure 2, verb infin. <**dobj**> and infin. <**dobj**>, respectively. The relevance statistics performed in the second stage will decide which one of the two extraction patterns is more representative of the domain.

The process of Stage 1 is summarized in Figure 3.

2.5.2 Stage 2

In this stage, we examine both relevant and irrelevant texts to compute the relevance statistics for each pattern. For each pattern in our dictionary, we go through each sentence in both our relevant and irrelevant corpora and keep track of the number of cases that were activated by this pattern. We use this to estimate the conditional probably that a text is relevant given that it activates a given extraction pattern according to the formula:

$$P_r(rel\text{-}text \mid text\ contains\ pattern_i) = \frac{rel\text{-}freq_i}{total\text{-}freq_i} \quad (1)$$

where $rel\text{-}freq_i$ is the number of times $pattern_i$ appeared in the relevant corpus while $total\text{-}freq_i$ is $rel\text{-}freq_i + the\ number\ of\ times\ pattern_i$ appeared in the irrelevant text. The idea behind computing the conditional probability for each pattern is that domain-specific expressions will show up more in relevant texts than irrelevant ones. Since AutoSlog-TS generates thousands of extraction patterns, we need to rank each pattern in order of relevance to the domain and discard the less important ones. This will allow for a person to review the most relevant patterns. The ranking function used by AutoSlog-TS is a simple one:

$$rank_i = relevance\ rate_i * log_2(total\text{-}freq) \quad (2)$$

where $relevance\ rate_i$ is as calculated by $eq(1)$ and $total\text{-}freq$ is also the same as in $eq(1)$. Rank is set to 0 if $relevance\ rate_i$ is ≤ 0.5. $Eq(2)$ gives a higher rank for patterns with a high frequency. This was done to save patterns that are common in both relevant and irrelevant texts from poor ranking. Both equations 1 and 2 were originally used in AutoSlog-TS. Riloff admits that they are simple and may be improved upon but they were suitable for her purposes. Even though we feel similarly, strictly improving AutoSlog-TS was not the focus of our work, so we decided to use the equations as presented.

The process of Stage 2 is summarized in Figure 4.

2.6 Rule Clustering & Generalization

When AutoSlog-TS outputs its rule extraction patterns and sorts them according to the ranking function, there is a danger that some important extraction patterns might be discarded. For example, in our problem domain of choice, the phrase "<Rookie Ali Curtis > netted <his second goal>" might have been ranked a lot higher than "<Forward Chris Brown> notched <a goal>". If our rule representation relied solely on the words it found in the training text and their ranks, these would be treated as separate rules throughout the program and the second phrase may be discarded as irrelevant if the top x rules are blindly selected.

However, MAXIM keeps all the rules from AutoSlog-TS, computes the *POS-classification-vectors* for both noun phrases (i.e. the pre- and post- fillers) and the *filler-set* for each rule and runs a two-level clustering program. This program first clusters the rules with the same *filler-set* together. It then calculates the average *POS-classification-vectors* of these simple clusters ($simple\text{-}cluster_i$ to $simple\text{-}cluster_n$) and computes the cosine similarity between all vectors using two n x n matrices (one for the pre-filler slot and the other for the post-filler). Next it chooses the pair of simple clusters that are most related by finding the pair whose pre-filler and post-filler cosine similarities' sum is highest as long as the pre-filler similarity's value is above a set threshold and the post-filler similarity's value is above a separate threshold.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 84–91
Computer Science Department, Swarthmore College

Pre-filler:	Filler:	Post-filler:
1) *max-len*	1) *filler-set*	1) *max-len*
2) *POS-classification-vector*		2) *POS-classification-vector*
3) *required-POS*		3) *required-POS*

Figure 5: MAXIM Rule Generalization

Then it continues this process, not considering the previously paired simple clusters, finding the next most related pair of simple clusters until either all clusters are paired or there are no two clusters with both pre- and post-filler similarities higher than their respective thresholds. The second stage of the clustering program can be repeated, cutting down the number of clusters by half each time.

Once all the rules from AutoSlog-TS are clustered in their respective group, a human will go through each cluster and link the pre- and post-fillers to the appropriate slot(s) in the template to be filled. Irrelevant clusters are eliminated at this point and the good clusters are assigned template slots and fed in to our rule generalization program. This is the only phase that requires human involvement. We timed ourselves doing this task together on 628 simple clusters and it took us just under one hour. Compared with the 5 hours it took the AutoSlog-TS team to manually assign slots to approximately 2,000 rules and you find that we are already saving time, and this was before the second stage of the clustering, where we cut those 628 simple clusters into less than 314 clusters.

The generalization program is very similar to our clustering algorithm since it relies heavily on POS information. As discussed in section 2.2, the three constraints for the pre and post filler slots are *max-len*, *POS-classification-vector*, and *required-POS*, and the only constraint for the filler slot is the *filler-set*. The *POS-classification-vector* for each cluster is computed by just taking the average of the individual *POS-classification-vectors*. From this vector, we obtain the *required-POS* by finding the maximum element. When the rules are applied to a test set, a sentence has to satisfy a total of four constraints before the content of its pre- and post-filler slots are sent to the template slot as important information. The structure of our rule representation is summarized in Figure 5.

2.7 Discussion of Results

We set out to fill a template with five slots: the name of the teams that played, the winner of the game, the score, the scorer(s), and the names of people who were ejected from the game. MAXIM allows for multiple values per slot using a comma as the delimiter. Ideally, the slots are filled with only the relevant information. However, the slots are generally filled with entire noun phrases, which contain the desired information (as the example output in Figure 6 shows). Note that this is still better than some research-level systems, which return the entire sentence that contains the desired information.

Figure 6: A Filled Template

soccer−game template

teams: The Colorado Rapids, the Los Angeles Galaxy 1 - 0
score: the Los Angeles Galaxy 1 - 0
winner: The Colorado Rapids
scorer: Agogo
ejected: Galaxy defender Ezra Hendrickson

	WSJ		Brown	
	Recall	Precision	Recall	Precision
1^{st} 25% of corpus	19.85%	68.97%	19.54%	65.31%
2^{nd} 25% of corpus	10.53%	61.33%	11.91%	64.2%

Table 1: Effect of Writing Style and Irrelevant Text Choice on Performance

Although our training corpus comprises 1,300 soccer articles (800,000 tokens), it was not possible to train our implementation of AutoSlog-TS within the given time frame. As a result, we trained on only 1/4 of our corpus size. This made it possible to analyze the effect of different writing styles within our problem domain on the performance of our system, as we could pick different parts of our corpus that correspond to the different soccer leagues. We tested MAXIM on 200 articles and calculated recall and precision by comparing the templates filled out by MAXIM to a human-filled answer key. As can be seen in Table 1, both recall and precision were better when we trained on the first 25% section (section A) of the training corpus than the second 25% section (section B). This is believed to be due to the fact that this second section contained mostly articles about Premier League games while the test corpus contained only articles from the 2000 MLS season. The writing styles from the British writers and the American writers varied greatly. For example, where one British writer writes, "<player> dinked a delightful cross", the average American writer writes, "<player> blasted a cross".

In addition, the result of different choices of irrelevant text was analyzed by training our system on both the Wall Street Journal (WSJ) and the Brown corpora[4]. We were hoping to show that training on the WSJ corpus would lead to better results than training on the Brown due to the commonality of the journalistic writing style between

[4]Of course, the size of these corpora is proportional to that of the relevant text.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 84–91
Computer Science Department, Swarthmore College

		WSJ		Brown	
		Recall	Prec.	Recall	Prec.
1^{st} 25%	*team*	24.74%	72.39%	23.47%	74.80%
	score	11.86%	95.83%	11.86%	92.00%
	winner	15.34%	83.33%	7.98%	86.67%
	scorer	21.26%	60.00%	23.56%	55.91%
	ejected	10.26%	100.0%	12.82%	55.56%
2^{nd} 25%	*team*	9.44%	52.11%	12.50%	61.25%
	score	4.64%	81.82%	5.15%	83.33%
	winner	7.36%	75.00%	7.98%	68.42%
	scorer	15.13%	62.70%	15.90%	63.36%
	ejected	2.56%	100.0%	2.56%	100.0%

Table 2: Breakdown of Results According to Slots

	team	*scorer*	*winner*	*score*	*ejected*
Rules	73	39	12	7	4
Recall	24.74%	21.26%	15.34%	11.86%	10.26%
Prec.	72.39%	60.00%	83.33%	95.83%	100.0%

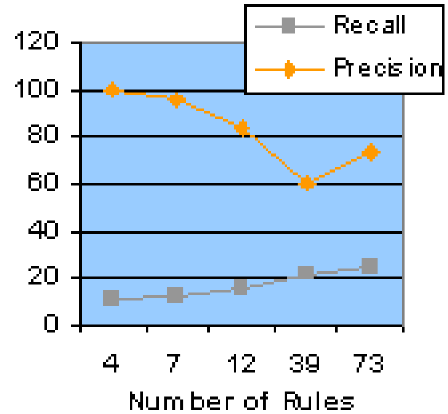
Table 3: Number of rules assigned to each slot in the Section A/WSJ rule-set compared to both the recall and precision for the slot.

this corpus and our relevant training corpus. We thought that a group of common journalistic phrases may appear a lot in the relevant training corpus but hardly at all in the Brown and thus be deemed as relevant. The results are inconclusive. We have not been able to link any effects to the choice of irrelevant text.

The breakdown of our results according to the five slots in our template is shown in Table 2. Our recall was generally best for the *team* and *scorer* slots. This is true in our results from section A as seen in the first half of Table 2. These recall values are followed, in descending order, by *winner*, *score*, *ejected*. These numbers correspond exactly with the number of rules that were assigned to these slots as clearly shown in Table 3. It is not surprising that the more rules assigned to a slot, the higher the recall for that slot, as there are a greater number of ways to fill the slot.

The precision values are ordered in exactly the opposite way (with the exception of the *scorer* slot) as is also seen in Figure 7. This is inversely related because the greater the number of rules, the greater the possibility of a mistake. Also, precision is dependent on how specific a rule is to the desired information and how commonly it is used in the problem domain. For instance, the most common rule to fill the *scorer* slot often appears something like, ”Diallo scored in the 17th minute”. However, it is also very common to read, ”D.C. United scored in the 17th minute”. Despite the presence of this second form, we must assign this rule to the *scorer* slot since this is the most likely way that we will find this information. Here

Figure 7: Recall/Precision as Functions of the Number of Rules



		WSJ		Brown	
		Recall	Prec.	Recall	Prec.
Stage 2 done once	1^{st} 25%	19.85%	68.97%	19.54%	65.31%
	2^{nd} 25%	10.53%	61.33%	11.91%	64.20%
Stage 2 done twice	1^{st} 25%	10.84%	79.44%	4.78%	74.12%
	2^{nd} 25%	2.65%	64.81%	4.62%	74.39%

Table 4: Clustering vs. Recall and Precision

we are sacrificing precision in order to increase recall.[5] We found that these kinds of decisions were required frequently. Unfortunately, we were not able to experiment enough to become adept at choosing the best choice. We believe that this, poor choice of rules to keep/delete and slots to assign them to, played a role in our less-than-ideal results.

The effect of the number of times the clustering algorithm was run on the performance of our system was also examined. The second stage of our clustering program reduces the number of clusters by half every time it is run, cutting the human involvement time by the same amount. However, this is done at the cost of recall, as shown in Table 4. The most likely reason for this is that more good rules are discarded because they were grouped with three other irrelevant rules. This decreases the number of rules assigned to each slot, which, as was seen in Figure 7, directly influences recall. The over-all increase in precision when clustering again can be explained by the same logic, though it is somewhat counterintuitive. One may very well expect that precision would decrease with increased clustering. This was not our experience, though we still expect that this behavior may be seen if one were to cluster more than the two times that we did.

[5]Examples like these are suspected to be the reason for the low precision of the *scorer* slot because they are very common, thus the anomaly in Figure 7.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 84–91
Computer Science Department, Swarthmore College

What was reassuring were the results of the clustering. After clustering twice, with nothing but part-of-speech information and *filler-set* information, we were left with less than $1/4$ of the "rules" that we started with. [6] Of these "rules", or rather, clusters of rules, some very different *filler-sets* were quite correctly grouped together. Some of these include {*blanked, defeated, rocked, rattled*}, {*knotted, narrowed, had scored, pressed*}, {*netted, notched, missed, scored in*}, and {*was issued, was shown, assumed, was ejected for*}. Obviously, "missed" and "assumed" do not belong in their respective clusters, and "pressed" is certainly arguable at best, but the rest of these results are fairly remarkable for the simplicity of the system in its current state.

Besides the potential improvements discussed in Section 2.8 that could be added to improve MAXIM, there were some issues that could be held accountable for the low recall and unexceptional precision. First, in the problem domain we were dealing with, figurative/descriptive language was used heavily in order to bring the excitement of the game to the article. This results in the same idea being expressed in a multitude of ways and with a variety of action words. We could not adequately train for this since it just took too long. Resorting to training on $1/4$ of our corpus hurt us here. This is not as big a problem for the domains many other IE systems have chosen to test on.

Secondly, we found it quite common for authors of an article to refer to the previous week's games, or other games from earlier in the season. So if MAXIM finds "<player> scored 3 goals" it has no way of determining whether or not this happened during the current game or a previous one and must assume the former. We automatically get this wrong if the phrase in question is not describing the current game.

Another problem we are sure that we ran into to some degree is human error in both generating the answer keys and comparing our results to the answer keys. For one thing, any of the problems that MAXIM runs into when extracting information from an article, a human runs into as well. Also, there is always question about what to do about things like own goals in soccer. Who to mark as the scorer is likely up to debate and this results in inconsistencies.

The last major problem that we had was errors from pre-processing. Some abbreviations escaped detection by our sentence boundary disambiguator. Words like "give-and-go" and "game-winner" were split into multiple parts and the hyphens were chunked as being outside of the surrounding noun-phrase by fnTBL. This broke up some noun-phrases that would have otherwise held extractable information. Finally, and mistakes made by the POS tagger directly effected our performance.

[6] We had less than $1/4$ of the rules because the clustering algorithm discards any rules that are not similar enough to another rule.

2.8 Conclusions

As it is clearly seen in the Results section, MAXIM suffered from low recall. There are several factors that might be responsible for this. Working towards the elimination or minimization of these factors, we believe, will improve both recall and precision.

The ability to train on the whole corpus may improve recall as more rules will be detected. This will also avoid the writing style dependency discussed in the Results section. If MAXIM trains on MLS, Premier League, and World Cup stories (the whole corpus), it will have a more generic and style-insensitive rule dictionary.

In addition, we might have been too restrictive and inflexible with our AutoSlog-TS's heuristic rules. The addition of rules to Figure 2 that are targeted to our problem domain and the elimination of those which are unlikely to be useful will result in a more refined AutoSlog-TS output. We also required our rules to be in *¡subj¿ verb ¡dobj¿* format in order to make the two phases of MAXIM compatible (see Section 2.2). However, if we allow *¡subj¿ verb* and *verb ¡dobj¿* to exist by themselves (which means that we have to allow for empty pre- or post- fillers), we could improve our recall.

Furthermore, we would like to minimize the exact-word-matching dependency of the filler slot. The implementation of this phase was considered using WordNet but preliminary trials indicated this to be a futile effort. For example, WordNet will not consider {*blanked, defeated, rattled, rocked*} to be in the same semantic class. Even though these words were grouped together by our clustering algorithm to form the *filler-set* for the *winner* and *team* slots, MAXIM will depend on finding one of these words in the test set during the application of the generalized rules. The sentence *¡Team A¿ crushed ¡Team B¿*, for example, will not help us extract *team* and *winner* information. The use of a thesaurus or ontology would have been ideal if, but they just do not currently have a rich enough synonym base to be of any practical, real-world use in any specific problem domain. At least not one where figurative language is used as often as in the domain we trained and tested on. It is worth noting that when Rapier incorporated WordNet senses in its rules, its performance was not significantly better than without this information.

Incorporating position information in our *POS-classification-vectors* might improve our clustering. Currently, we only keep track of the frequency of each POS tag within a noun phrase. If this can be extended to include some kind of contextual information, e.g. the position of POS tags or POS tag n-grams, we may be able to more accurately group different clusters expressing the

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 84–91
Computer Science Department, Swarthmore College

same idea together. This would both decrease the human involvement required, by decreasing the amount of attention necessary when examining clusters, and increase precision, by minimizing the number of irrelevant rules in any cluster.

Faults aside, we believe we have presented a novel approach to information extraction that requires less human preparation and supervision than any system available at this time. With further research into how to most effectively translate AutoSlog's rules to MAXIM's/Rapier's format recall can be greatly increased. Similarly, more work looking into a comprehensive list of extraction patterns for AutoSlog-TS would be of great benefit. Improved relevancy-rate and ranking functions for AutoSlog-TS would help to get good rules to the top of the list.[7] As mentioned above, an adequate thesaurus/ontology would be of the greatest benefit when it comes to generalization. Also, an improvement on the representation of the *POS-classification-vector* would be helpful. With improvements made in these areas, MAXIM may prove to be as effective as other systems available while requiring at most half as much human involvement. This is a big step towards the development of IE systems practical for real-world, multi-domain use.

References

Mary E. Califf and Raymond J. Mooney. 1997. Relational Learning of Pattern-Match Rules for Information Extraction. In *Proceedings of the ACL Workshop on Natural Language Learning*, pages 9-15. AAAI-Press, Menlo Park, CA.

G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. 1993. Introduction to WordNet: An on-line lexical database. Available by ftp to clarity.princeton.edu.

G. Ngai and R. Florian. 2001. Transformation-based learning in the fast lane. In *Proceedings of NAACL'01*, pages 40-47.

E. Riloff. 1993. Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 811-816. AAAI Press/MIT Press.

E. Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044-1049. The AAAI Press/MIT Press.

[7] Although we were able to abandon looking at the ranking of rules altogether. The clustering was so effective and time-saving that we sent all of the rules AutoSlog-TS outputted into the clustering stage and just deleted bad clusters, taking care of multiple rules at once.

Appeared in: *Proceedings of the Class of 2003 Senior Conference*, pages 84–91
Computer Science Department, Swarthmore College