

Boosting Relation Extraction Recall With Soft Rules

Jeffrey P. Bigham

May 13, 2005

Abstract

Many efficient and scalable methods for statistical information extraction from large corpora are based on the use of simple extraction patterns. Unfortunately, these rigid patterns have limits in terms of recall that are not acceptable when extracting members of higher-arity relations. We present a method that attempts to counter this drawback with flexible, on-the-fly rule learning that enhances traditional rigid extraction patterns with a set of softer constraints. By allowing rule learning and extraction to occur simultaneously, the system is able to efficiently leverage contextual clues when evaluating extraction and rule quality. We show that such rules increase recall at the cost of precision and explore several assessment methods in an attempt to compensate for the decreased precision.

1 Introduction

The web represents a vast collection of information, spread among billions of web pages and expressed in a variety of ways. While advanced information retrieval techniques empower humans to find information of interest with a reasonable amount of effort, amassing large collections of facts (useful for issuing powerful queries or for performing inference) remains quite daunting. Many recent systems attempt to automate this task by learning and applying simple contextual patterns designed to extract facts from textual corpora. Such systems use these extraction patterns as low-level workhorses on which higher levels of assessment and inference can be applied. In this paper, we seek a robust method for both learning and applying such extraction rules and introduce a novel technique in the context of KnowItAll[8] that combines pattern learning with pattern application. To that end, we introduce a new type of extraction rule that incorporates generalized wildcards as soft constraints and allows for an automated rule assessment component.

As a motivating example, suppose we want to learn how birth state influences one's chances of someday becoming the President of the United States (only 19 states can claim a U.S. President and, of those, 4 states account for over half of all U.S. Presidents). Given a table of all U.S. Presidents along with their birth state, such inference is straightforward. KnowItAll relies on extraction rules for learning the elements of this table (see Figure 1). However, the current system exhibits several shortcomings that this paper hopes to address.

The base system is quite adept at bootstrapping rules capable of extracting instances of a particular class. These satisfy the `instanceOf(Class, Instance)` predicate which expresses the relation that `Instance` is of type `Class`. For example, if the predicate's first argument is bound to *city*, then *Seattle* and *Beijing* are both valid bindings of `Instance`. The pattern “`cities such`

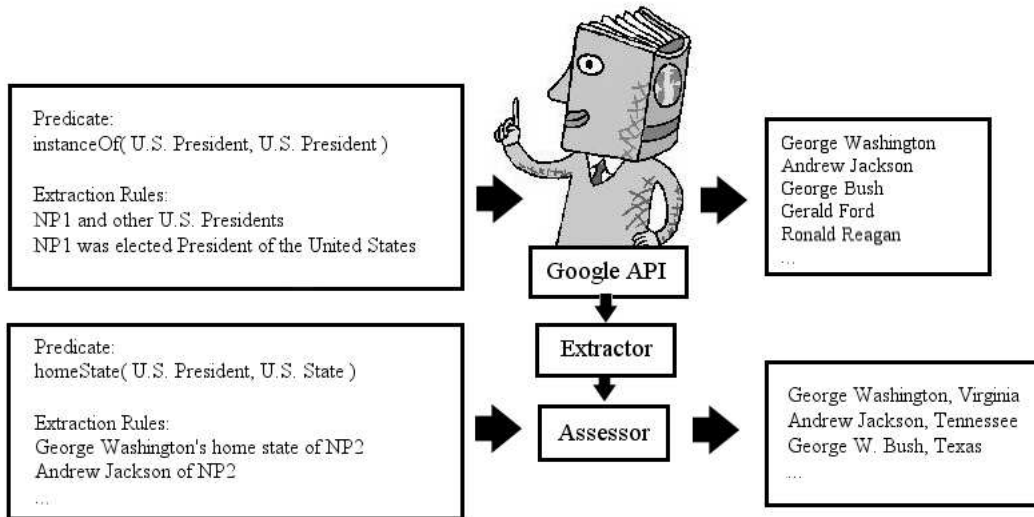


Figure 1: KnowItAll Learning Presidential Home States. First, KnowItAll generates instances of the class “U.S. President” by applying extraction rules to webpages returned by a Google query and then assesses the likelihood that each is true. Next, it uses those generated presidents to bind the first argument of the binary extraction rules and uses them to extract and then to assess instances of the “home state” relation.

as NP1” is a high recall, high precision extraction rule for generating members of the class *city* from text. In this rule, the text labelled NP1 is a slot that can be bound to a noun phrase to form a new extraction only when seen along with the remaining context of the rule. In this paper, we explore the difficulties that arise when extending this method to generalized relations between two arguments, such as `mayorOf(City, Mayor)` and `starsIn(Movie, Actor)`. Extending these methods to this more general case has, in our experience, been more difficult. The biggest hurdle to overcome is that such relations are mentioned much less frequently on the web, which means our extraction methods must become more tolerant to variety in how relations are expressed. In this paper, we will refer to extractions generated for the `instanceOf` predicate as unary extractions generated by unary extraction rules. Along the same vein, we refer to extractions generated for predicates with arity two as binary extractions generated by binary extraction rules.

Extraction rules designed to generate unary extraction generally follow the template of the original Hearst patterns and can be readily extended with rule-learning[7]. Extraction patterns for binary relations tend to be more domain specific and generally achieve much lower levels of recall than their unary counterparts. This follows directly from [7] in which it was noted that binary extraction rules targeted at a variety of semantically different target attributes make excellent unary extraction rules and, therefore, are often a subset of them. The intuitive idea behind this is expressed in Table 1. As an example, the patterns “mayor of NP1” and “NP1’s neighborhoods” are high-quality extraction rules for the predicate `instanceOf(Class, City)`. Syntactic variation in expressing these relationships may cause the system to miss a few cities discernable by the stated semantic relationship. For instance, neither preceding rule will match “Seattle’s mayor attended”

nor “Brooklyn is a neighborhood of **New York**,” but given enough rules with decent coverage over a variety of semantic or syntactic relationships, we are likely to extract a missed city by another rule. In binary extraction, we have both fewer semantic variations to explore and fewer instances over which to observe stated relationship and, therefore, direct our effort at leveraging the full spectrum of syntactic variation in how each relation is expressed. In general, we expect the rules capable of making valid extractions to follow Zipf’s Law[2], meaning that few rules will generate most of the desired extractions, but to extract the rest, we must consider a large number of low recall rules. We expect not to be able to gather such low recall rules adequately before extraction time and, as such, extracting binary relations with high recall presents a difficult problem that we have largely been able to ignore in the past.

Web Hit Counts For Cities and City Mayor Pairs			
City	Hit Count	City + Mayor	Hit Count
Seattle	86,100,000	Seattle “Greg Nickels”	24,600
Cleveland	35,200,000	Cleveland “Jane Campbell”	17,300
Beijing	22,600,000	Beijing “Wang Qishan”	7,490
Liverpool	15,100,000	Liverpool “Frank Roderick”	299
Yakima	3,980,000	Yakima “Paul George”	508
Sendai	977,000	Sendai “Hajimu Fujii”	82
Pickerington	296,000	Pickerington “Dave Shaver”	18

Table 1: Web Hit Count For Cities vs. City-Mayor Pairs. Hit Count returned by Google for queries for a selection of cities vs. hit counts for queries for each city and its current mayor. On average only .02% as many pages contain both the mayor and the city as compared to those which contain at least the city. This suggests that binary extraction rules must be more robust because they have fewer pages to they can be applied relative to the unary case.

The modifications to the KnowItAll system presented in this paper offer potential solutions for the problems outlined above and we evaluate their effectiveness on several realistic extraction tasks. The new system learns binary extraction rules automatically, in a scalable fashion, requiring no user intervention to generate instances of new classes other than an initial extraction pattern of the type several systems already produce [1, 7]. Flexible “soft rules” allow the system to learn new extraction rules implicitly at extraction time and a new assessor component rates the confidence in each new extraction rule automatically based on contextual information unique to each extraction.

2 Overview Of System

The process of soft extraction takes as input a predicate and a hard rule of the type generated by a number of different systems[16, 7, 14, 13, 3]. It then softens the given rule by adorning a portion of the chosen context with wildcards via simple heuristics, and finally, it applies that rule to text found on the web. The wildcards are similar to those used in regular expressions, but unlike those used in regular expressions, they are not greedy. The wildcards can be expanded favoring strings with the shortest edit distance from the given hard rule. During the extraction process, the tagged sentence and the location of the extraction within it are stored along with the extraction. This textual information combined with other features, such as the edit distance between the given

hard rule and the applied soft rule, form the contextual representation that our new contextual information assessment will leverage in an attempt to boost the precision/recall trade-off of soft rules.

Our system implements three methods of assessment for extractions generated by soft rules: frequency-based assessment, Pointwise Mutual Information - Information Retrieval (PMI-IR[17]) assessment, and Contextual Extraction Assessment (CEA). Frequency-based assessment assigns a higher likelihood to those extractions observed most often. In our system, extractions observed in identical sentences are counted only once. Our version of PMI-IR assessment proceeds assuming we have high confidence in both the bound first argument and the relation and assigns a probability to each extraction based solely on an evaluation of the second argument. This process was already in the core KnowItAll system and is explained fully in [8]. CEA proceeds by first building an AdaBoost classifier over various labelled and unlabelled data sets (described in Section 6). New extractions are assigned an approximate probability of being correct or incorrect (described below) and these probabilities are used either in a filter-based approach or as input to a noisy-or evaluator. Each method has a natural threshold and, by varying this threshold, we control the precision/recall tradeoff observed in the system.

In the following sections, we discuss soft extraction and contextual extraction assessment in depth.

3 Soft Extraction

3.1 Extraction Pattern Creation and Application

The inputs to the soft extraction system are an existing binary extraction pattern and a list of instances for the first argument of the binary relation. This base extraction pattern is modified by a heuristic to generate the soft version of the extraction pattern. If the label of the second argument’s class appears in the original extraction pattern in any form, it is marked as static and cannot be removed during rule application. All remaining portions of the contextual pattern are marked as *soft*, allowing each to be removed or expanded during rule application. During the extraction process, soft patterns are expanded and contracted in order of increasing word-level edit distance (as originally described in [11]), favoring extractions for which the total edit distance is smallest and allowing a maximum total edit distance of 6.

As an example, consider the pattern “**NP2**, mayor of **NP1**” used to extract instances of the `mayorOf(City, Mayor)` predicate. Expressed as a regular expression, the soft extraction equivalent is “**NP2**,* mayor of* **NP1**”, meaning that both the comma and “of” in the pattern text may be replaced and expanded. Thus, the instantiated version of this rule “**NP2**,* mayor of* **Seattle**” may now match “**Greg Nickels** was recently appointed mayor of **Seattle**” despite the phrase “was recently appointed” not appearing in the original rule and the comma not being found in the extraction. The edit distance score of this match is 4 because of the deletion of the comma and the insertion of the 3 words in the phrase “was recently appointed.”

3.2 Query Generation and Ordering

To efficiently find webpages to which a hard extraction pattern might successfully be applied, we use the Google API and use the substrings of the pattern that are set before application as keywords. Continuing from the example above, the instantiated extraction rule “**NP2**, mayor of

Seattle” would generate “mayor of Seattle” as its sole keyword phrase sent to Google. To generate keyword phrases for soft rules, we follow the process just described, although for efficiency reasons we explore the keyword phrases in decreasing order of constraint. The soft rule first generates the same keyword phrase generated for the hard version of the rule, “mayor of Seattle.” It then relaxes the “of” constraint, because it is labeled “soft”, leading to a second query consisting of the phrases “mayor” and “Seattle.” Normally, additional rules would have been generated by relaxing the “,” context, but Google does not recognize punctuation in its searches.

In the experiments presented in Section 6, patterns are explored uniformly to the same depth for every generated query, but in practice the system would likely be altered to explore soft rules either when it has decided that it has not found enough extractions using standard hard rules or after it has exhausted hard rule extractions.

4 Contextual Extraction Assessment

The extraction assessment process assigns a probability to each unique extraction made by the system. In past versions of KnowItAll, individual extractions were assessed by learning Pointwise Mutual Information - Information Retrieval (PMI-IR) thresholds over a set of discriminators [4, 17] and combining such features using Naive Bayes and, more recently, by modeling the likelihood of receiving each extraction via a frequency-based assessment method [6]. Neither model incorporated a confidence in extractions based on the rule with which they were extracted largely because a robust estimate of each rule’s quality is not available in their framework.¹ CEA changes this by collecting a set of contextual features associated with each extraction on which the quality of each extraction rule can be estimated. Because the evaluation of each matching rule is done automatically at extraction time, even rules that match only one extraction can be included and properly evaluated. CEA provides a method for estimating the quality of extractions generated by rules that were not seen in training, and this ability to robustly extract facts from a wide variety of syntactic variations forms the chief advantage of soft rules.

CEA proceeds by first building and training a supervised classifier and then using it to assign a confidence to each extraction rule. Each extraction is represented as a large number of features designed to represent the context in which each was found and to express the variation between the particular instantiation of the soft rule used for extraction and the input hard rule. Intuitively, we expect extraction rules that match very closely to the input rule to be more likely to be correct, because the closer they are to the original rule, the more confident we are that the expanded rule did not change the original meaning of the rule. Conversely, inserting words such as “not” or “and” or introducing commas or parentheses to the rule are likely to change the meaning of the rule and, therefore, should lower our confidence. The list of potential feature types and a demonstration of their application is shown in Table 2.

We experimented with two variants: one that simply removed individual extractions deemed more likely than not to be incorrect before assessment and one that used the binomial model to combine hypotheses generated for each individual extraction by CEA to form a final probability for each final unique extraction. To rate the confidence in a particular unique extraction, this method considers probabilities assigned by all classifiers and rates the likelihood that all are false. Formally, given n probabilities p_i assigned by the CEA classifier to each of n extractions e_i that evaluate to

¹The Urns Model, introduced in [6], did use the number of particular types of rules which generated the same unique extraction as one of its parameters.

Possible Features With Application	
Extraction Rule: <i>NP1</i> , mayor of <i>NP2</i>	
Sentence: The _{DT} assassination _{NN} of _{IN} Mr Daniel, who _{WP} was _{VBD} also _{RB} mayor of Santo Andre , one _{CD} of _{IN} the _{DT} satellite _{NN} cities _{NNS} of _{IN} greater _{JJR} Sao _{NNP} Paulo _{NNP} ...	
Feature Type	Match or Value In Sentence
5 Words Before	The assassination of
5 Parts of Speech Before	DT NN IN
Word Before	of
Part of Speech Before	IN
In Extraction Word Additions	who was also
In Extraction Part of Speech Additions	WP VBD RB
Word After	,
Part of Speech After	,
5 Words After	, one of the satellite
5 Parts of Speech After	, CD IN DT NN
Edit Distance	3
Capitalized Word Added To Extraction	false
Dictionary Word in Argument 2	false
Length Argument 2 (Words)	2

Table 2: The Feature Space. This table lists the space of possible features used by the classifier and shows the result of each as extracted by the given rule for the `mayorOf(City, Mayor)` predicate.

the same unique extraction, the noisy-OR function assigns a final probability P for the unique extraction E as follows:

$$P(EisCorrect) = 1.0 - \prod_i p_i \quad (1)$$

If thought of as a weighting function, this computation gives high weight to extractions that are observed often and assigned high probability by the classifier.

4.1 Classification

For classification, we use the AdaBoost[9] meta-learner using pruned C4.5 Decision Trees[12] as the base learner. Both are implemented in the Weka[18] machine learning package. AdaBoost is just one of a set of meta-learners shown to iteratively improve the performance of a number of learning algorithms by concentrating on increasingly harder examples during each round.

4.2 Feature Selection

Feature selection is automatic in the system. First, all features that appear in a random sample of the training data are collected and then the top thirty are chosen based on the information gain of each feature with respect to the class over the training data. This is identical to the metric used by

the C4.5 Decision Tree algorithm in determining which attribute is best to split on at each level. It is defined in terms of the entropy:

$$Entropy(S) = \sum_{c \in C} \frac{-p(c)}{\log_2(c)} \quad (2)$$

$$Gain(S, A) = Entropy(S) - \sum_{a \in A} \frac{S_v}{S} \times Entropy(S_v) \quad (3)$$

Top 20 Features Chosen For mayorOf(City, Mayor) Predicate		
Feature Value	Feature Type	Information Gain
Edit Distance	NA	0.25
(In Extraction Part of Speech	0.09
(In Extraction Word	0.09
)	After Part of Speeches	0.09
)	After Words	0.09
.	After Words	0.08
.	After Word	0.07
.	After Part of Speech	0.07
.	After Part of Speeches	0.07
,	In Extraction	0.05
,	In Extraction Part of Speech	0.05
)	After Part of Speech	0.05
)	After Word	0.05
Dictionary Word In Arg2	NA	0.04
of	After Words	0.03
in	After Part of Speeches	0.02
of	Before Words	0.02
CC	In Extraction Part of Speech	0.02
VBD	After Part of Speech	0.02
Length of Arg2	NA	0.02

Table 3: Top 20 Features Chosen for mayorOf(City, Mayor) Classifier Trained on Extractions Generated for Different Predicates.

The goal of our feature selection process is not to pick the absolute best features but to reduce the initial space of possible features from a few thousand to a smaller subset that is both more manageable and more general. In our experiments, we chose thirty features according to this metric as we observed empirically during initial tests, that beyond this point, the features became quite dependent on the particular predicates used during training. Features chosen for the `mayorOf(City, Mayor)` predicate using predicate-independent training data is shown in Table 3. Features chosen for the `starsIn(Actor, Movie)` predicate using predicate-dependent data is shown in Table 4. In general, we observed that better features, in terms of classifier performance, were learned when training on extractions generated for that predicate. Currently our classifiers require hand-labelled training data. However, since KnowItAll seeks to be scalable and autonomous, we concentrate on using classifiers that are built from predicate-independent data.

Top 20 Features Chosen For starsIn(Actor, Movie) Predicate		
Feature Value	Feature Type	Information Gain
Dictionary Word in Arg2	NA	0.12
director	Before Words	0.07
:	Before Word	0.07
:	Before Words	0.06
NNP	Before Part of Speeches	0.06
.	After Words	0.05
:	Before Part of Speeches	0.05
:	In Extraction	0.05
:	Before Part of Speech	0.05
.	After Part of Speeches	0.03
VBG	Before Part of Speeches	0.03
by	Before Word	0.03
:	In Extraction Part of Speech	0.03
by	Before Words	0.02
NN	Before Part of Speeches	0.02
“ Before Word 0.02		
VBG	Before Part of Speech	0.02
of	After Word	0.02
directed	Before Words	0.02
NN	Before Part of Speech	0.02

Table 4: Top 20 Features Chosen for starsIn(Actor, Movie) Classifier Trained on Extractions From Same Predicate.

5 Background and Related Work

A main advantage and challenge presented by KnowItAll, with respect to its predecessors, is the ability to build from unary extraction and assume a set of assigned instances for one of the arguments in the relation to be extracted. Many existing systems[1, 15] assume that either the arguments or the relation is known before extraction and tackle the problems simultaneously. This has potential drawbacks in terms of directing the system toward a particular relation and in building a final table of facts that is densely populated. With fewer knowns, they can afford to be less forgiving of the particular syntax used in expressing each relation to be extracted. This differs from our goal in that we want to find relations even if they are only mentioned very rarely, which prompts the use of rules more flexible to syntactic variation.

SnowBall[1] generates and applies flexible extraction rules. It differs from our system in that it fully separates rule learning and extraction, and the rules produced are not as expressive as soft rules. It introduces a rule language that is flexible relative to hard rules and allows extraction rules to be relaxed during extraction but only in ways seen during training - in essence allowing each hard rule to expand into a number of hard rules seen during training, each with an assigned likelihood as observed during training. In contrast, our system learns generic rule guidelines that allow us to extract instances from patterns not seen during training. In their system, the extraction confidence is based on thresholding a cosine similarity metric. While not as powerful as the models we learned,

Summary of Related Work							
System	System Assumptions and Features						
	Known Rel.	Mult. Rel.	Relaxed R's	Unique R's	Parse	Supervised	Web
Soft Rules with CEA	Yes	Yes	Yes	Yes	No	No ²	Yes
Snowball[1]	No	Yes	Yes	No	No	No	No
Grishman <i>et al.</i> [10]	No	Yes	No	Yes	No	No	No
Snow <i>et al.</i> [15]	Yes	No	Yes	No	Yes	Yes	No
McCallum <i>et al.</i> [5]	Yes	Yes	Yes	Yes	No	Yes	No

Table 5: Summary of Related Work. The above table summarizes several related systems. It records for each system: 1) whether the relation to be extracted is known *a priori*, 2) whether the system works across on multiple relations, 3) whether rules can be relaxed during application, 4) whether unique rules not observed in training can generate valid extractions, 5) whether a full parse of the corpus is required, 6) whether new training data is required to learn new relations, and 7) was the system tested on the web.

by doing so they averted the need for negative training examples, a principle disadvantage of the contextual information method of assessment.

McCallum and Culotta [5] trained Conditional Random Fields for extraction type classification essentially using the surrounding text as features. They observed very high performance but trained their classifier on the same domain on which they tested. As already observed this is not an attractive option for a scalable, autonomous system such as KnowItAll. In Section 6 we report high accuracy when using classifiers trained on extractions by the same base rule and for the same predicate during both training and testing.

A number of papers deal with learning textual extraction patterns [16, 7, 14, 13, 3, 10, 15, 5, 1]. Our main difference is that we deal with binary extraction with one argument bound and we learn these in the context of KnowItAll, meaning that we strive to be efficient and scalable. In that context, we introduce a method of extraction that is tolerant to syntactic variation and yet does not require an expensive parse of the corpus. A more extensive comparison between our work and several systems that are most similar is presented in Table 5.

6 Experimental Results

The stated goal of using soft rules and additional assessment methods for extraction is to increase the recall of hard rules without substantially sacrificing their precision; in the following series of experiments, we test the hypothesis that the system implemented achieves that goal. To facilitate this experiment, approximately 300,000 extractions were made using both the hard and soft versions of the rules in Table 6 and labelled by hand as either good or bad and hard or soft. These rules were chosen because of their good empirical performance and for their heterogeneity in both argument types and rule construction.

In our experiments, the first argument is bound by the base KnowItAll system using bootstrapping. The `electeIn` predicate is bound to 60 presidents (43 US presidents plus 17 reasonable variations of their names - e.g. John F. Kennedy \rightarrow JFK), while the rest are bound to 100 values of their respective first argument classes for the reported experiments. To provide a fair comparison,

Predicate	Extraction Rule
<code>electedIn(U.S. President, Year)</code>	NP1 <i>head</i> (elected) in YEAR2
<code>populationOf(City, Population)</code>	population of NP1 is <i>head</i> (NUM2)
<code>mayorOf(City, Mayor)</code>	NP2 , <i>head</i> (mayor) of NP1
<code>homeStateOf(Senator, State)</code>	NP1 of NP2
<code>starsIn(Actor, Film)</code>	NP2 starring NP1

Table 6: Predicates and Associated Extraction Rules used for experimentation. In each example NP1 is the noun phrase bound to the first argument of each predicate and one of **NP2**, **YEAR2**, or **NUM2** is bound to the second. *head*(x) in a pattern indicates that x must be the head of the noun or verb phrase at that location for the pattern to match.

we allowed a maximum of 200 web queries for each keyword phrase generated by each extraction rule (maximum of 20,000 webpages per keyword phrase), although with only a few exceptions, the hard rules generally exhausted their queries before reaching this limit.

The efficiency of the rules is explored more deeply in Figure 2 which shows the number of queries vs. the number of new unique extractions. At the point, where the hard rule ends is where all queries generated specifically for hard rules are exhausted. After this point the yield goes down although soft rules still manage to pick up more and more unique correct extractions. More interestingly, the soft rules outperform hard rules on extracting correct extractions from webpages gathered with hard queries, suggesting a recall bonus for using soft rules even without relaxed queries.

6.1 Soft Extraction

We hypothesized that our soft extraction procedure would increase recall but decrease precision and Figure 3 demonstrates this trend in our implemented system. While the precision of the soft rules consistently remains below that of the hard rules, the overall recall of soft rules is greater, significantly so for all but the `mayorOf()` relation. Because of the great heterogeneity of the predicates we chose, each demonstrates a different response to soft rules. In general those rules that perform well using hard rules often fall in precision, whereas that do not, such as the `starsIn()` predicate, perform even better.

Nevertheless, a system that increases recall but causes precision to drop precipitously is not useful so it is important to note that results for all but the `stateOf()` predicate see a drop of 20-30%. This was expected and is on the order of what we had hoped assessment would be able to restore. The `stateOf()` predicate’s poor performance was expected as it is the sole extraction rule that does not contain the name of the second argument’s class as a fixed part of the rule. As a result, only the bound first argument grounds this rule to the correct relation when it is relaxed, resulting in the extraction of numerous relations involving U.S. Senators. This rule demonstrates a known weakness of our approach and was included to measure its significance and the power of our assessment methods to overcome very noisy initial extraction.

6.2 Contextual Extraction Assessment

In this section, we explore the usefulness of CEA. To do so, we evaluate the performance of a number of classifiers trained on various training data sets that can be divided into two class: a domain-independent and a domain-dependent set. The domain-independent data set for each predicate

Predicate	Classifier				
	Hard	Soft	Same Predicate	ED Only	No ED
<code>electedIn(U.S. President, Year)</code>	96.48%	57.19%	86.51%	48.55%	56.18%
<code>stateOf(Senator, State)</code>	43.32%	85.86%	88.10%	28.18%	38.19%
<code>mayorOf(City, Mayor)</code>	85.12%	65.43%	87.90%	63.64%	54.94%
<code>populationOf(City, Population)</code>	9.09%	46.15%	77.16%	44.38%	58.82%
<code>starsIn(Actor, Film)</code>	33.20%	57.80%	74.29%	60.60%	54.52%

Table 7: Raw Classifier Performance. These results show the overall classification accuracy of five variations of classifiers on extractions using the extraction rules for the predicates as described in Table 6. The **Hard** and **Soft** classifiers operated over extractions generated from hard and soft extraction rules respectively and were trained individually for each predicate using extractions made for the remaining four predicates. The **Same Predicate** classifier was trained and tested for each predicate on soft extractions made for that predicate in a 10-fold cross validation manner. The **ED Only** and **No ED** classifiers trained and tested similarly to **Soft** but their feature sets were restricted to include only the edit distance feature or not the edit distance feature respectively.

consists of a random sampling of extractions gathered for the remaining four predicates listed in Table 6. This data set is designed to test how general the classifiers learned are. Ideally, we could train a classifier once in this manner and use it to classify all new extractions. As each classifier will eventually be tested on examples from the predicate that did not contribute extractions for its training, splitting the data in this way amounts to a 5-fold cross validation.

The domain-dependent data sets are built for each predicate using extractions for that predicate. The extractions are divided into ten subsets based on the value of each’s bound first argument and the classifiers that will later be used for each set of ten are trained on a sampling of data from the remaining subsets. This amounts to a 10-fold cross validation.

Our first experiments measure the effectiveness of simply not including extractions that each classifier rates as being more likely than not to be incorrect. Simply discarding such “bad” extractions is an attractive option because it can be performed directly at extraction time and the later levels of KnowItAll assessment need never see these extractions. Those extractions that do “survive” this initial screening are assessed as usual, which should particularly affect the ranking scores of frequency-based assessment methods. In Figure 4, we present the results of this experiment over our five predicates and rules using both hard and soft rules. In general, the method increases the precision but at the cost of much of our recall gains.

The experiments testing our CEA noisy-or ranking function are presented in Figure 5. The results indicate that they perform as well and often better than the filtering method just considered. Because this is a ranking function the noisy-or method also avoids the drop in recall experienced with the filtering approach.

We observed that often the CEA methods perform poorly on predicates unlike those on which their classifiers were trained. For example, unlike the other predicates evaluated, we observed that the `populationOf()` predicate often produces correct extractions using rules with high edit distance and with added punctuation. CEA performs poorly when its classifiers were trained on training data gathered for other predicates because extractions for the predicates on which it was trained do not have this quality. The PMI-IR assessor also has problems evaluating extractions for certain predicates, but it errs for different reasons. For example, it performs poorly on the

`starsIn()` predicate in large part due to its tendency of confusing actors and movies. Such observations led to us exploring an assessor that used both the noisy-or and PMI-IR ranking in concert. To compute the combined ranking function, we once again employed the noisy-or function, this time on the probabilities returned by the original noisy-or and by the PMI-IR assessment method.

The results are shown in Figure 6.

Although for reasons of scalability, we prefer assessment methods that do not require domain-dependent labelled examples, we report our results using such data as training to show that like previous work in this area, we can perform quite well given this type of data. The results shown in Figure 7 show our results, which are (not surprisingly) much better. The results are particularly favorable compared to all over assessment methods for the `starsIn(Actor, Movie)` predicate. The core KnowItAll system performs poorly in assessing this predicate (Figure 3) because both frequency-based and PMI-IR assessment tend to confuse movies and directors. The two highest features automatically chosen for this predicate’s classifier include whether a dictionary word is in the extracted argument and whether the word “director” appears in the five words immediately before the extraction (see Table 4). Both of these features serve to differentiate movies from directors. Because our predicate-dependent classifier is able to discover these differences and, therefore, rate directors much lower, use of this classifier increases performance greatly.

6.3 Feature Analysis

We observed that the edit distance feature was consistently chosen in our models with much higher information gain relative to other potential features. To evaluate the importance that the edit distance feature had in the final classification performance we built two models, one with edit distance as its only feature and the other with the top thirty features minus edit distance. The performance of this classifier is shown in Figure 8.

If edit distance were the sole feature on which the classifier’s performance relied, then we could implement a single rule based on the edit distance that would perform just as well and avoid much of the complexity of Contextual Extraction Assessment. Table 7 shows that the classifiers using only the edit distance features generally have raw performance lower than classifiers trained on the full set. Not surprisingly, the performance of these classifiers follows the performance of the hard rule frequency-based method, as shown in Figure 8. For predicates, such as `starsIn(Actor, Movie)`, these results demonstrate that features other than the edit distance feature contribute to the performance and generality of our CEA classifiers. These results are shown in Figure 8.

6.4 Rule Analysis

We hypothesized that to get higher recall we must accept a wide variety of rules, and that, for some bound first arguments, rules that would not be observed during training would be required. Table 8 shows the number of rules that were generated and applied by our system compared with the total number of rules required in the ideal case. In three of the cases over half of the rules generated were required to get the recall that the system obtained. The number of rules is very high and, as such, it is very unlikely that a rule learning system could obtain enough high-precision rules to match these results, although a direct comparison is a topic for future work.

Finally, to grasp how soft rule performance varies with how “soft” the rules are, in Figure 9 we graphed the performance in terms of both recall and precision of rules at each value of edit

Predicate	Generated Rules	Required Rules
<code>electedIn(U.S. President, Year)</code>	275	205
<code>stateOf(Senator, State)</code>	5513	663
<code>mayorOf(City, Mayor)</code>	326	178
<code>populationOf(City, Population)</code>	189	124
<code>starsIn(Actor, Film)</code>	887	354

Table 8: Number of Required Rules vs. Total Number Examined. This table shows the difference between the number of rules that were actually applied as compared to the fewest possible rules required to achieve the same recall.

distance. In general the precision of the rules dropped as edit distance was increased. Much of our recall gains were seen with edit distances of 1 and 2. These results suggest that rules with a smaller allowable edit distance of 1 or 2 would be sufficient.

7 Conclusions and Future Work

Soft rules provide a relatively easy way of increasing the recall of extraction patterns; extracting useful relations will eventually require methods more accepting than those currently available. An important opportunity for future work is devising an assessment scheme capable of improving their performance. One possibility is extending our current best method, explained in [6], to explicitly model different extraction rules, each with an associated probability. Another is devising a way to introduce domain dependent training data into assessment, perhaps by using the top extractions from hard rules or estimating the best through an EM-like procedure.

The results of our experiments show that soft rules are able to increase recall in binary extraction, but that this recall gain comes at the cost of precision. Our contextual extraction assessment method combined with PMI-IR assessment was able to counter this precision loss better than the other methods we tested, but not consistently enough to be practical. When our assessment method was trained on extractions from the same predicate on which it was to be tested, as was done in much of the previous work, it was largely able to restore precision, but this is unattractive for a variety of reasons. Our results suggest the reward and a possible path toward future work capable of boosting the precision of soft extractions to the levels we have come to expect, in a domain independent and scalable fashion.

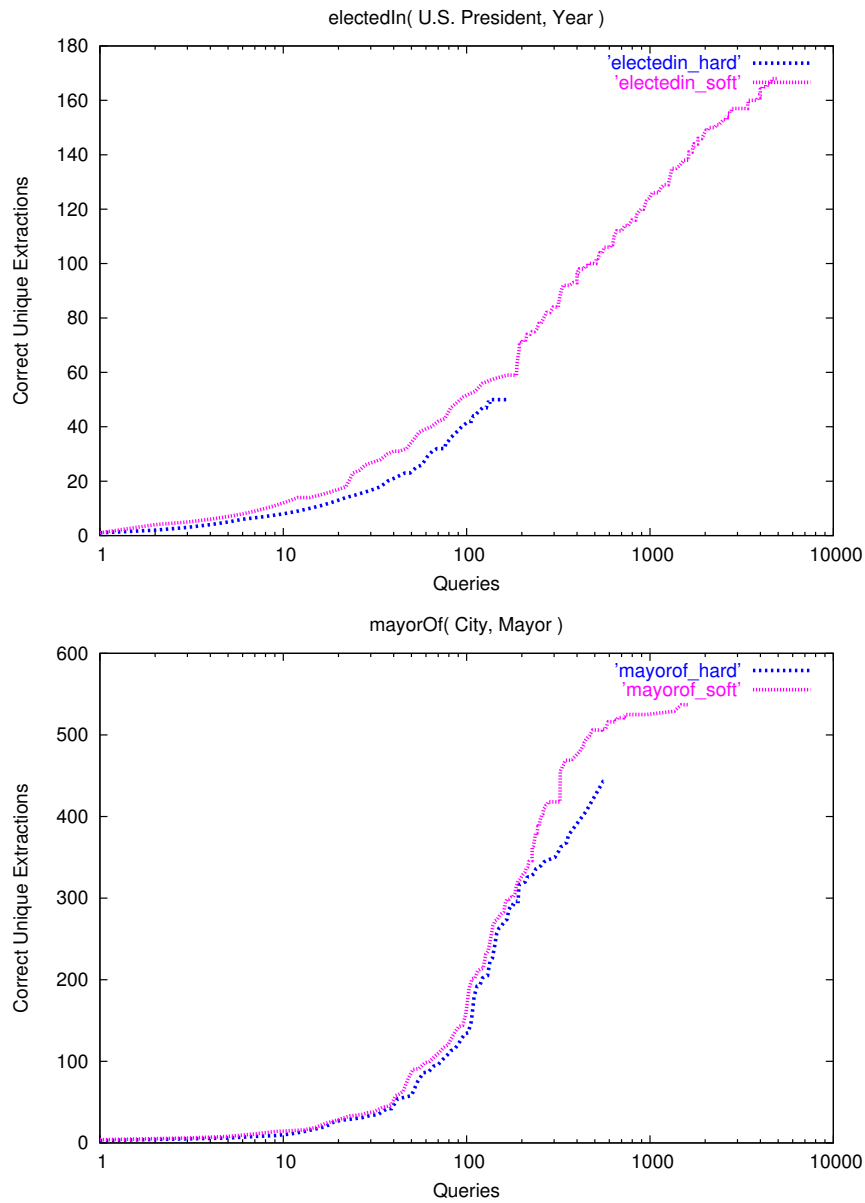


Figure 2: Queries vs. Unique Correct Extractions. These graphs explore the differences in yield of hard and soft rules. The queries are ordered firstly from most to least constrained keyword phrases and secondly by yield within queries for the same keyword phrase. The number of queries is expressed on a log scale as the yield for queries decreases dramatically as they become more generic. Most interesting is the increase in recall soft rules obtain on all tests using the same queries as hard rules, suggesting that the inefficiency associated with the low yield of the least-constrained soft queries may be avoidable by applying soft rules only to those pages acquired by the most-constrained queries. These results only show ideal recall, precision will be considered during experiments involving assessment. (continued next 2 pages)

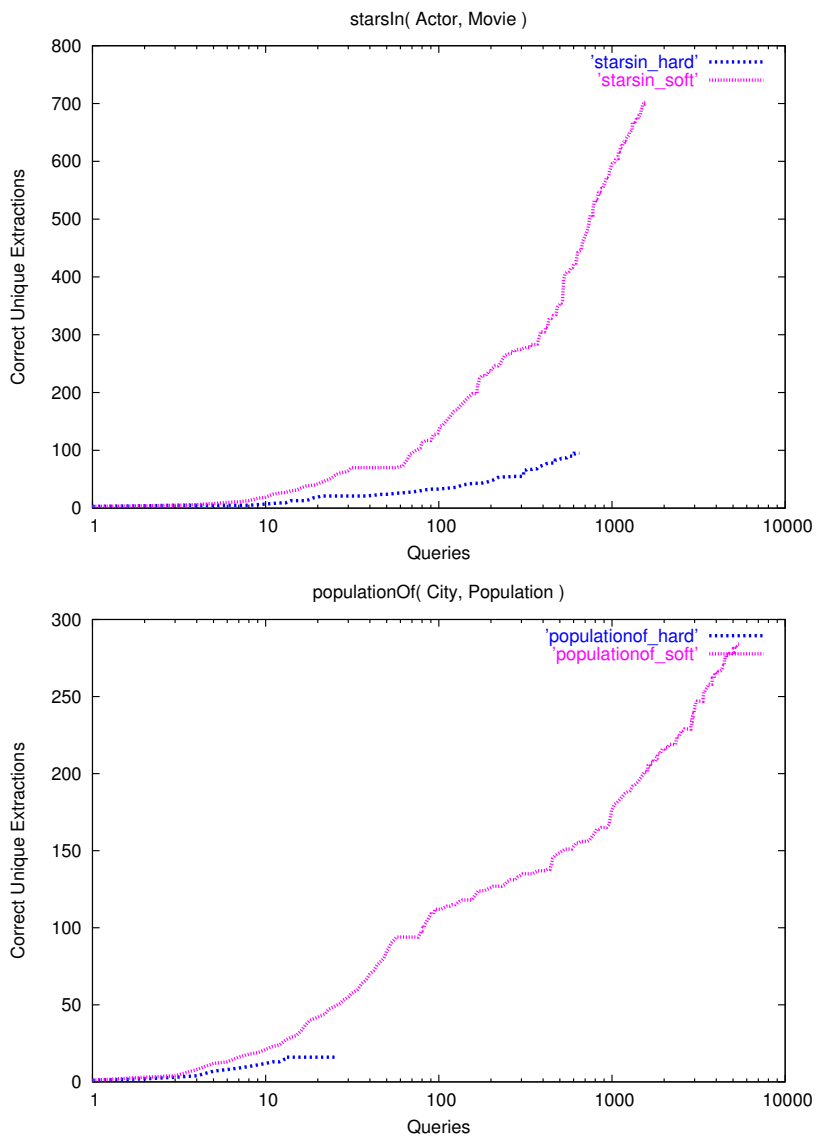


Figure 2: Queries vs. Unique Correct Extractions (continued)

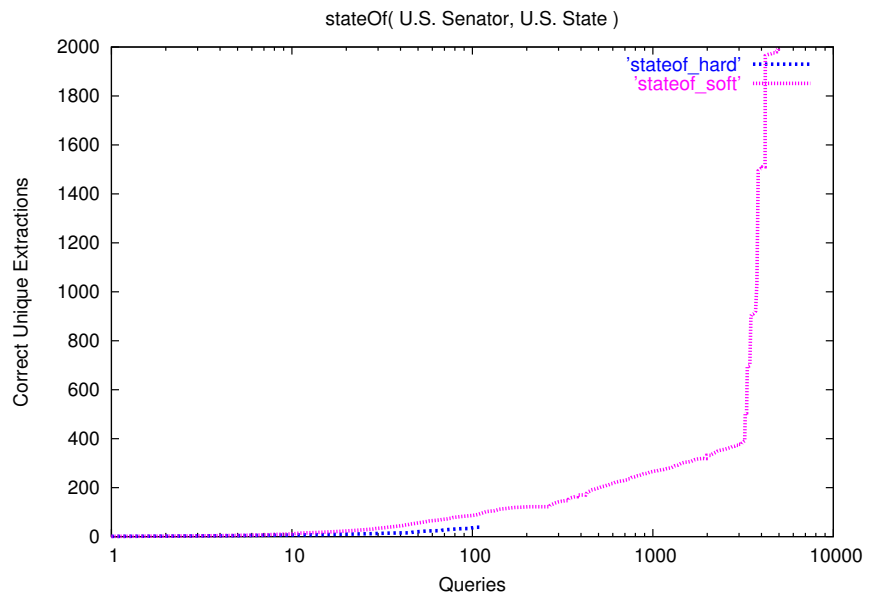


Figure 2: Queries vs. Unique Correct Extractions (continued)

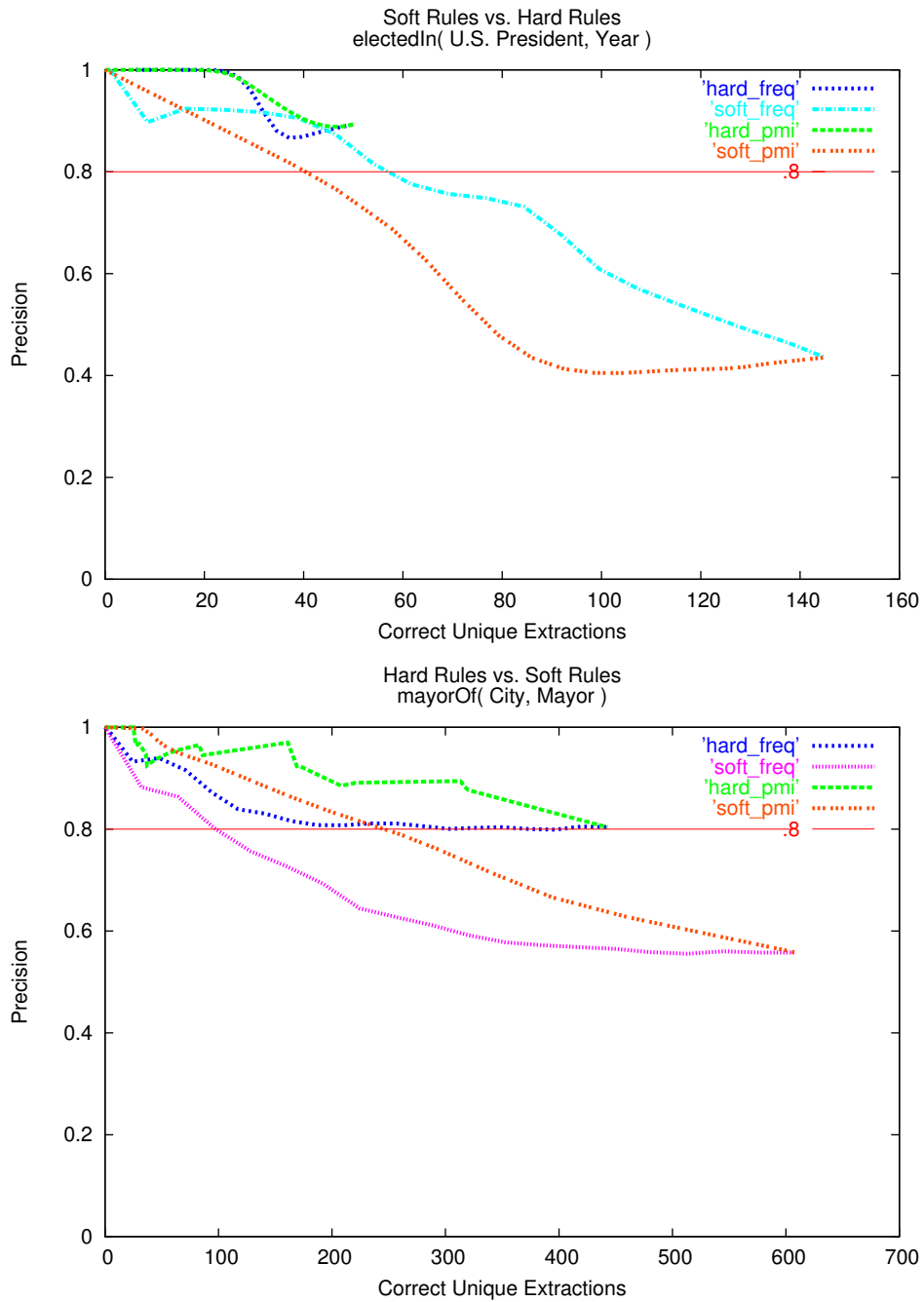


Figure 3: Soft Rules vs. Hard Rules: Plotted above is the performance of the basic soft rules as compared to the hard rules from which they were derived in terms of the number of unique correct extractions versus the precision of the system. Assessment is reported using both PMI-IR assessment and using a frequency-based metric that counts multiple extractions from identical sentences as one. (continued next 2 pages)

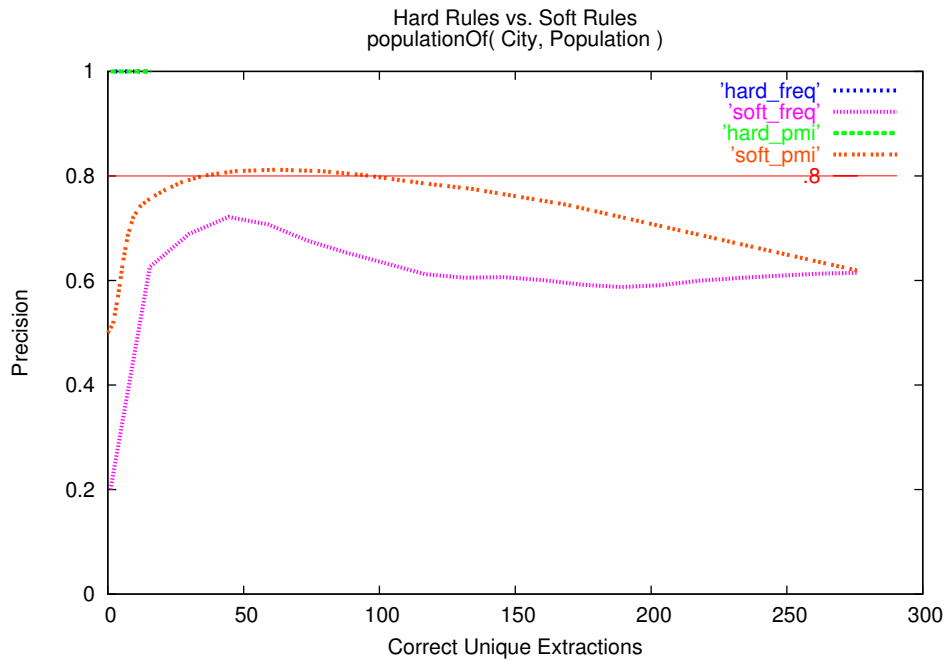
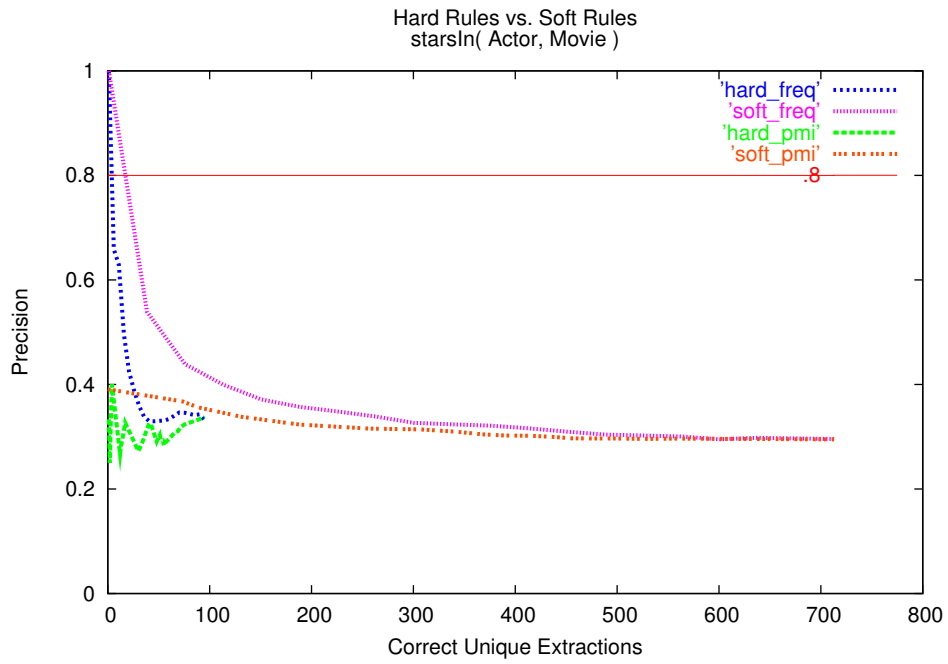


Figure 3: Soft Rules vs. Hard Rules (continued)

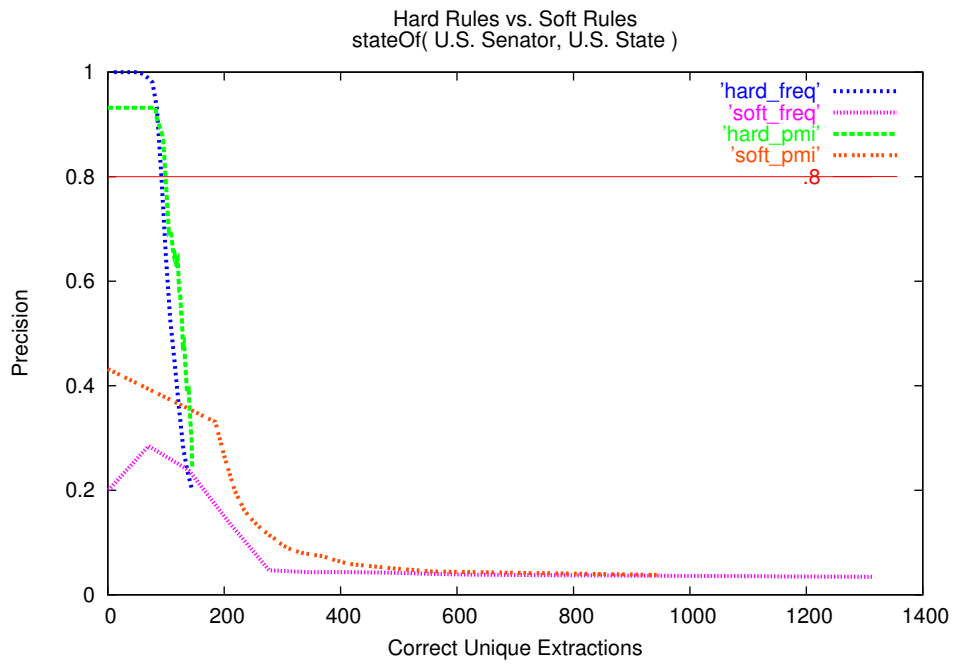


Figure 3: Soft Rules vs. Hard Rules (continued)

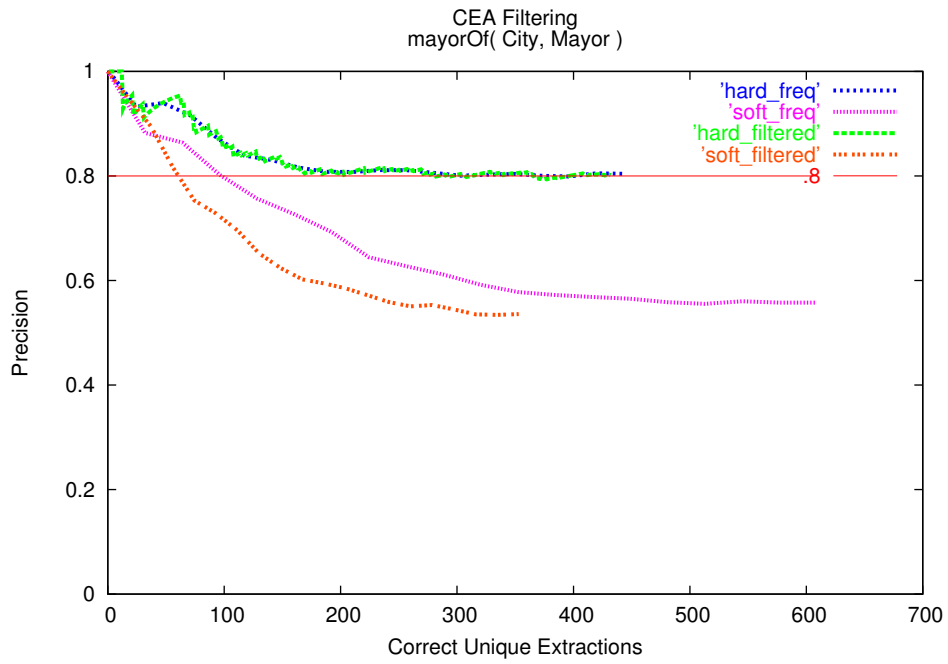
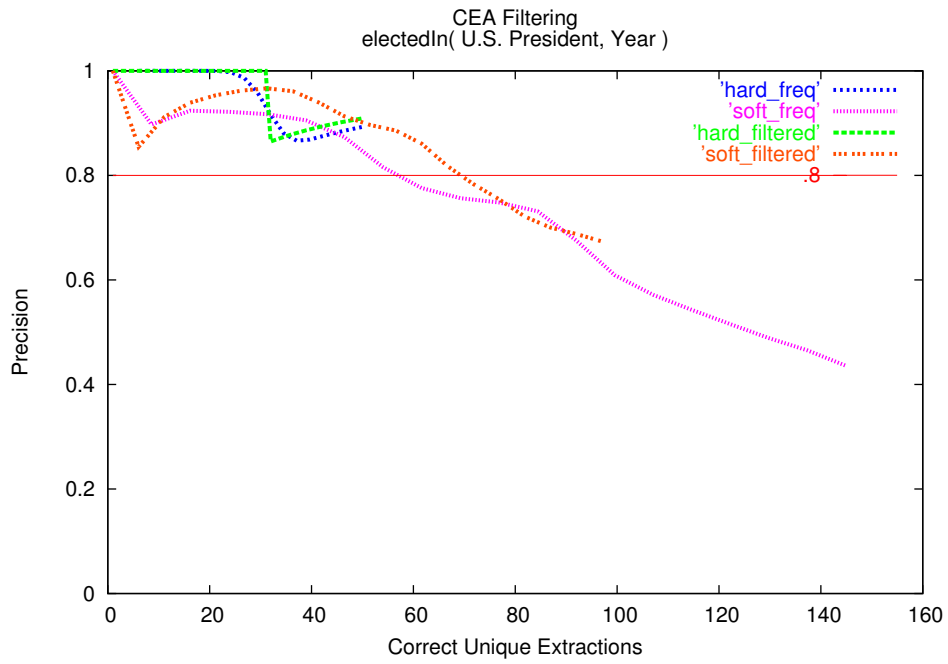


Figure 4: CEA as a filter. (continued on next 2 pages)

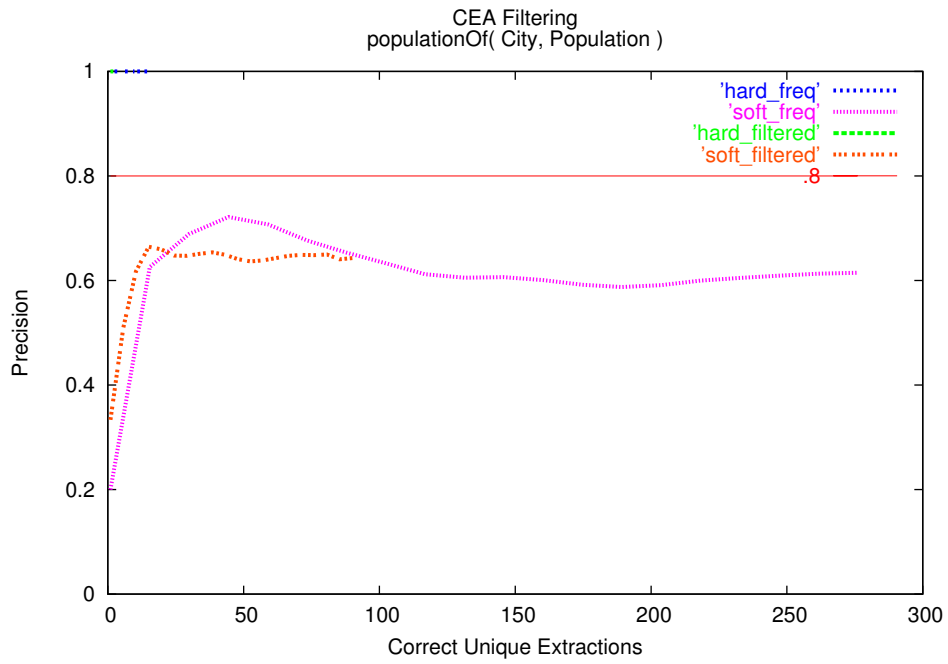
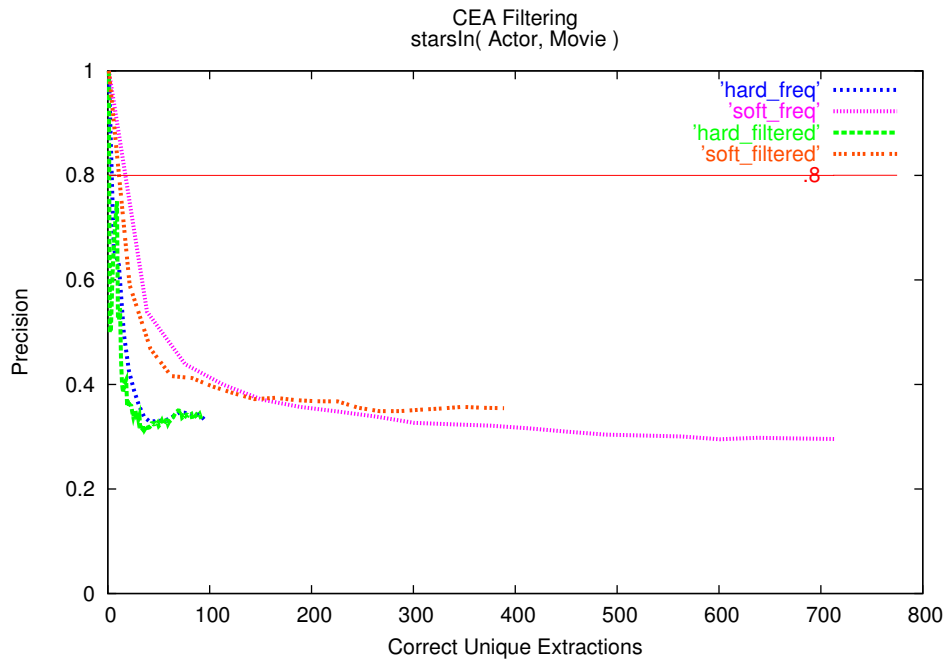


Figure 4: CEA as a filter. (continued)

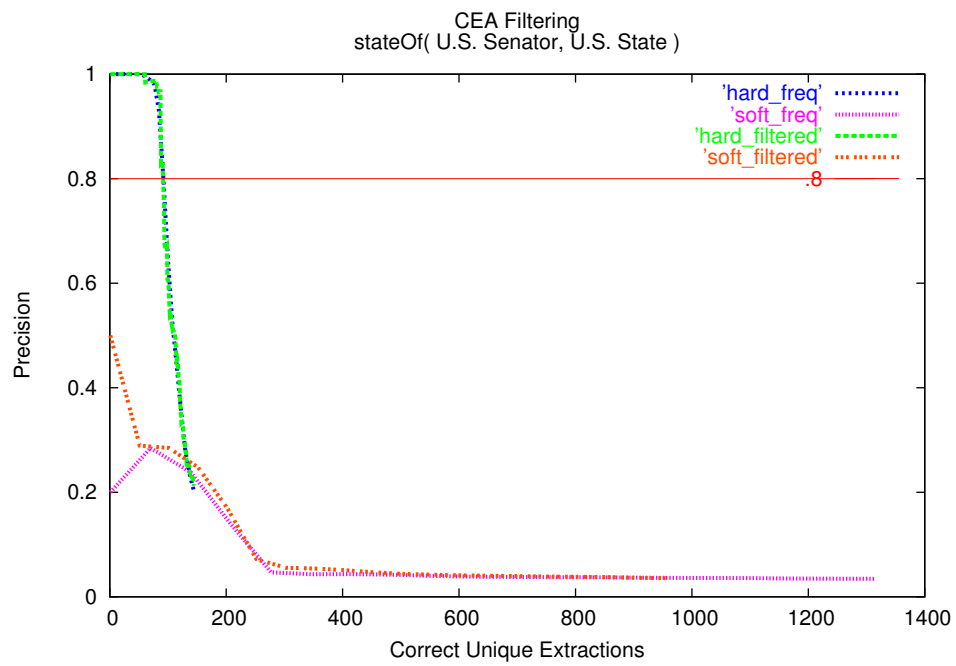


Figure 4: CEA as a filter. (continued)

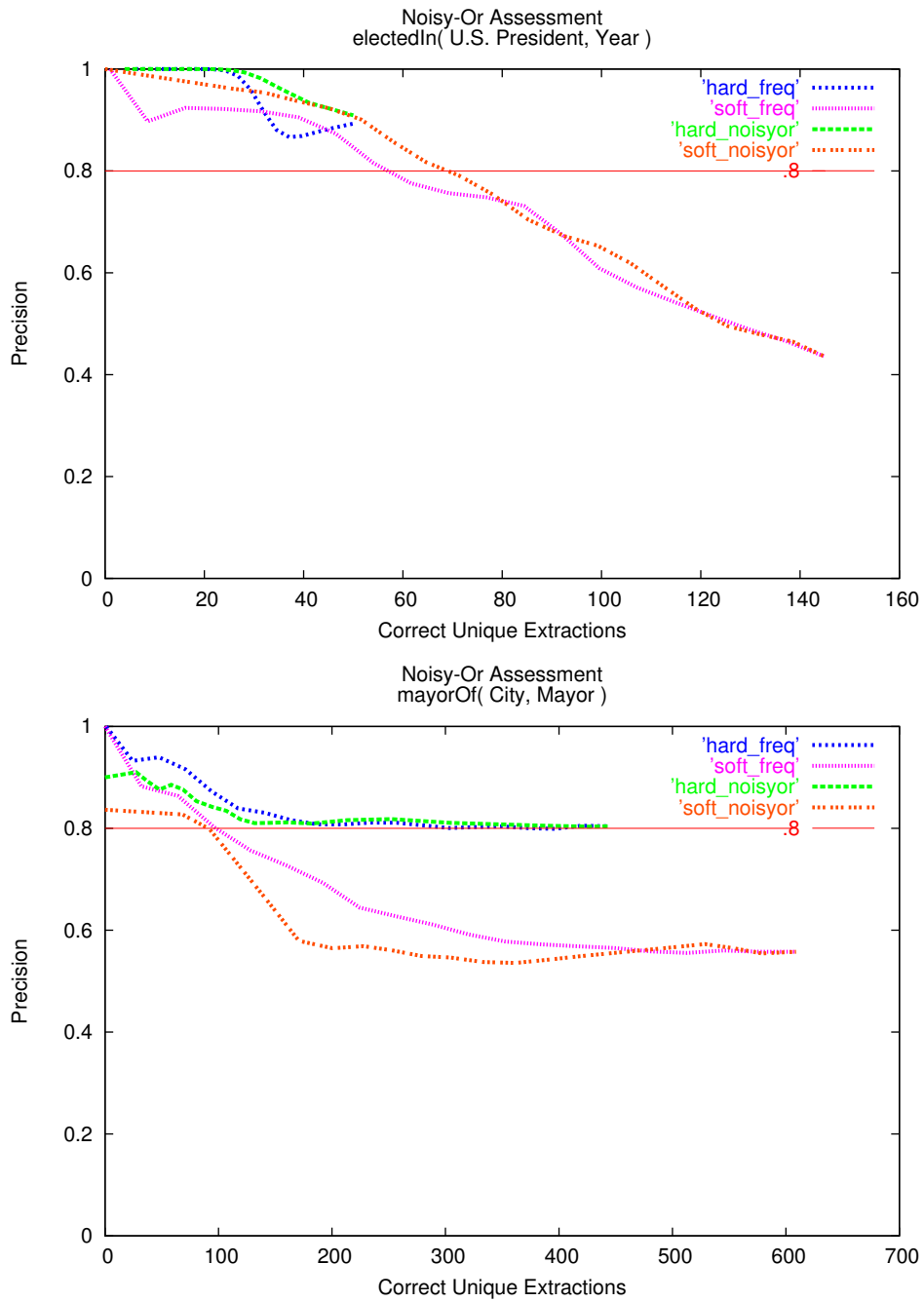


Figure 5: Noisy-Or Ranking of CEA probabilities (continued on next 2 pages)

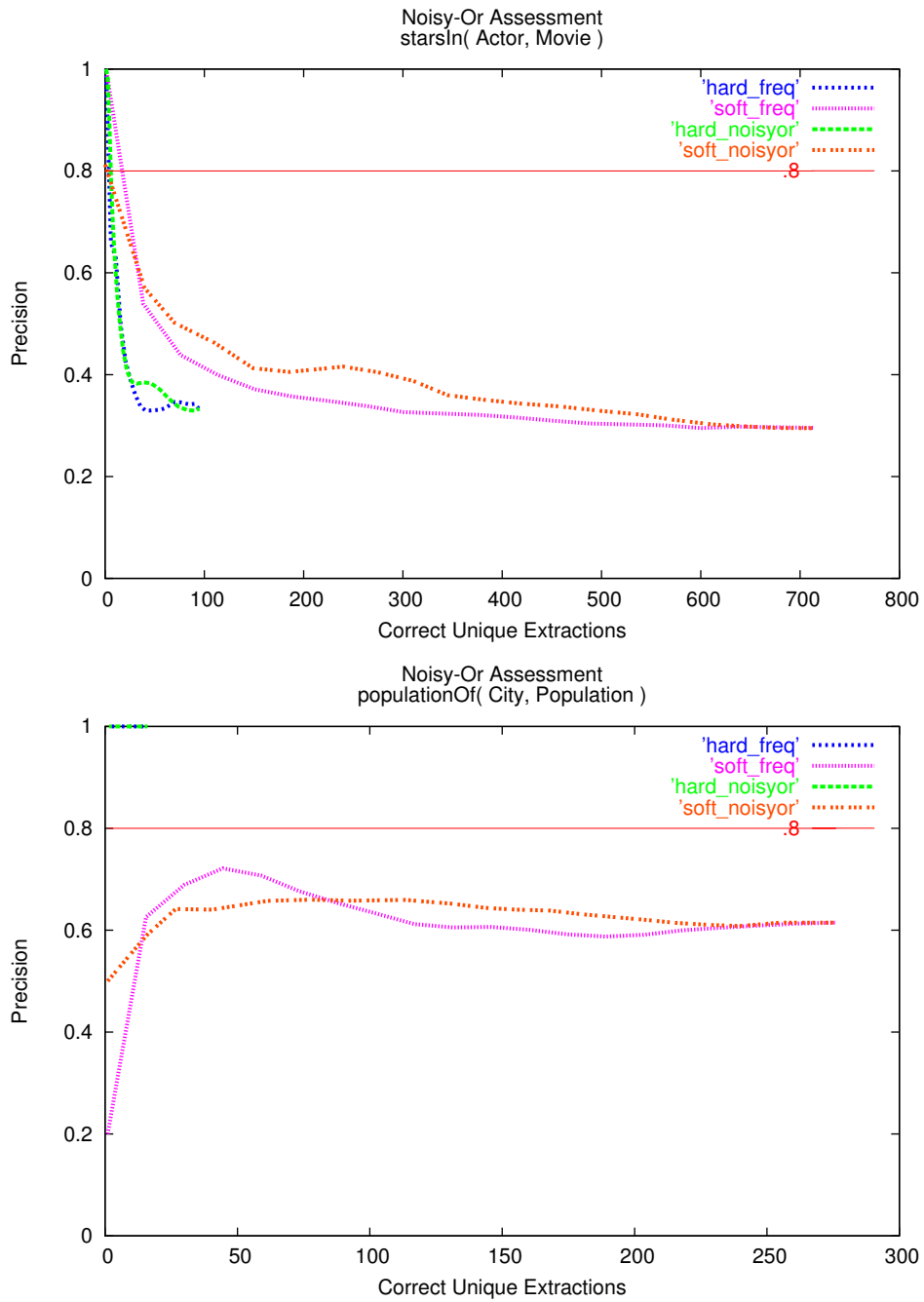


Figure 5: Noisy-Or Ranking of CEA probabilities (continued)

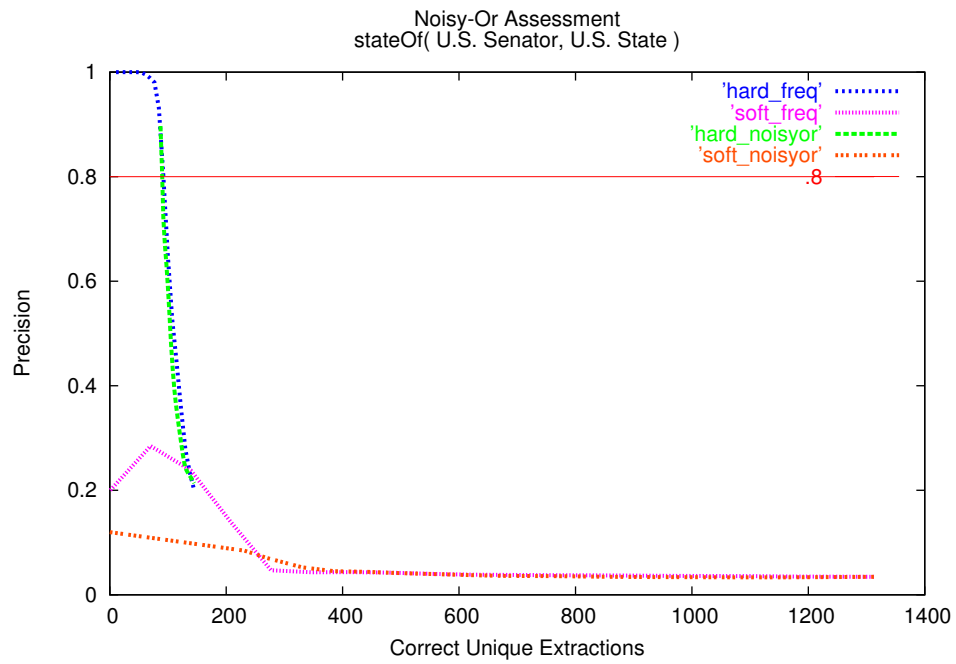


Figure 5: Noisy-Or Ranking of CEA probabilities (continued)

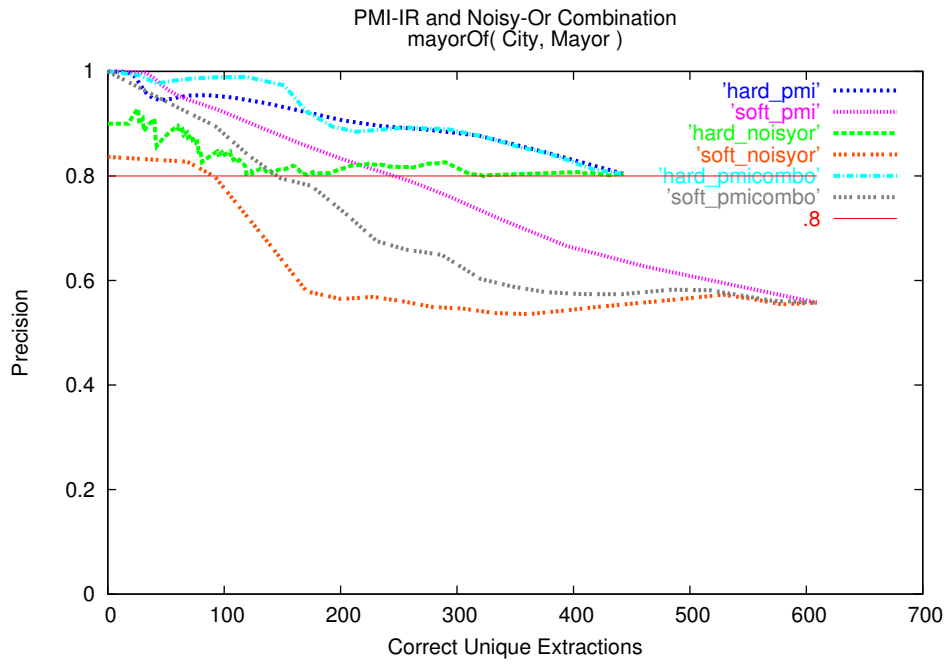
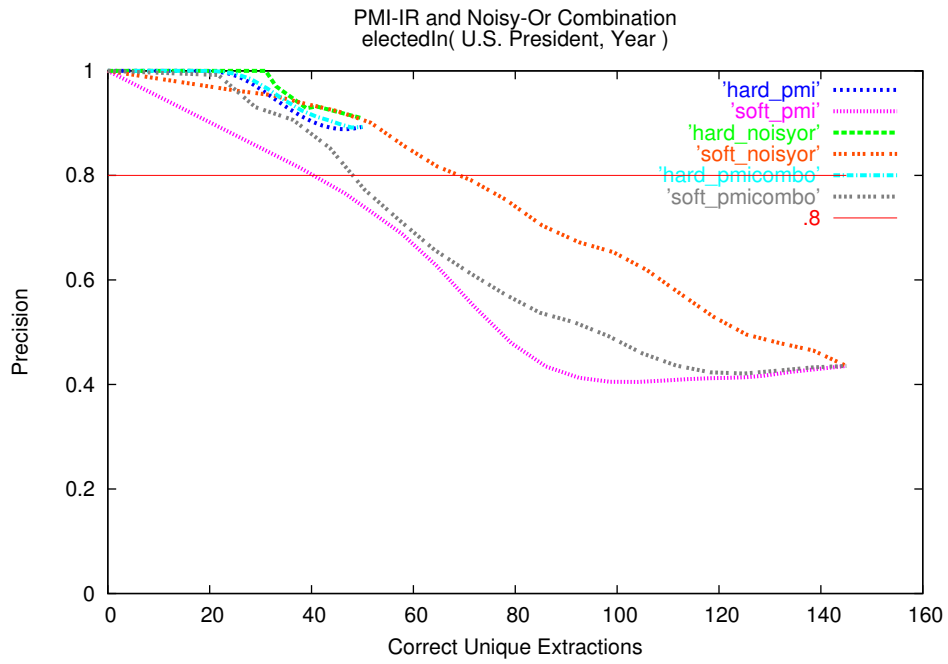


Figure 6: PMI-IR Combined with CEA (continued next 2 pages)

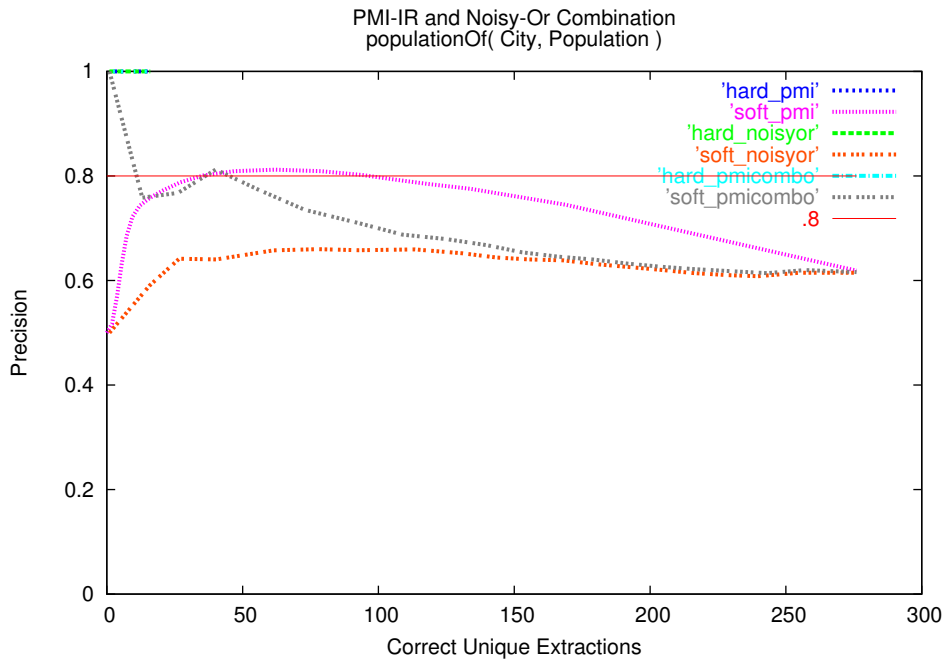
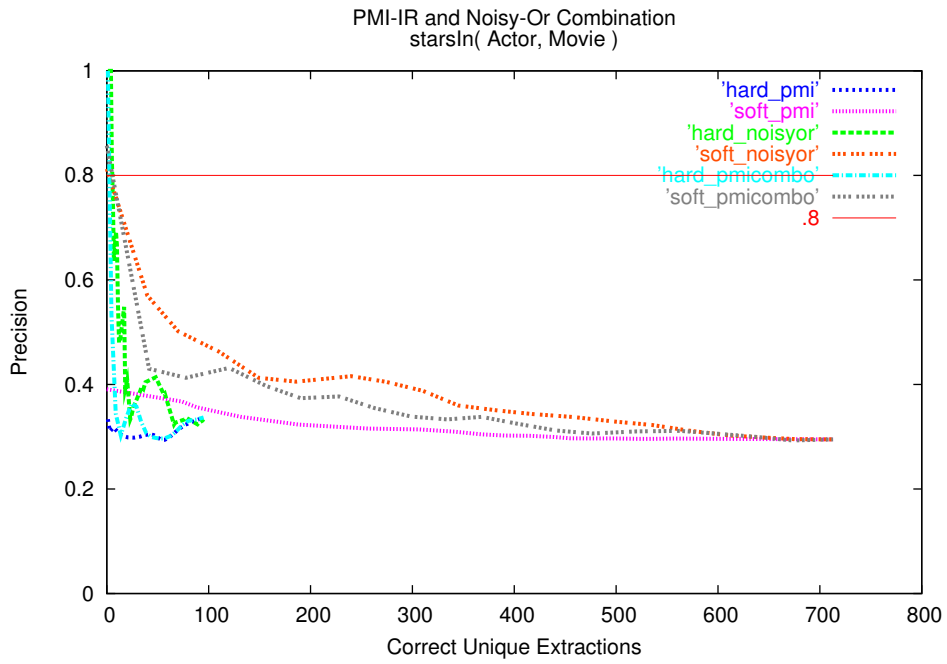


Figure 6: PMI-IR Combined with CEA (continued)

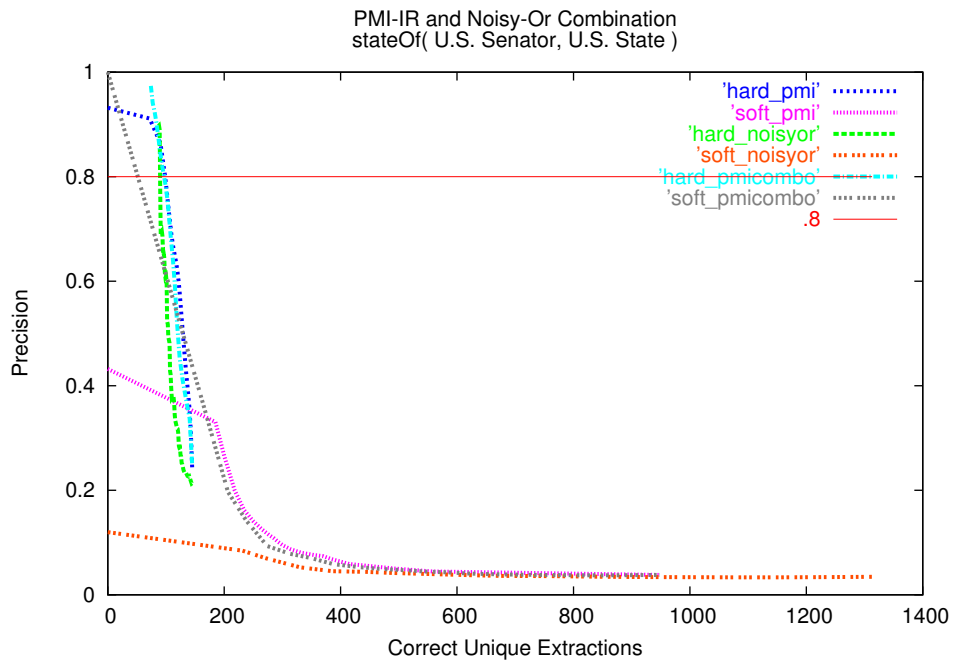


Figure 6: PMI-IR Combined with CEA (continued)

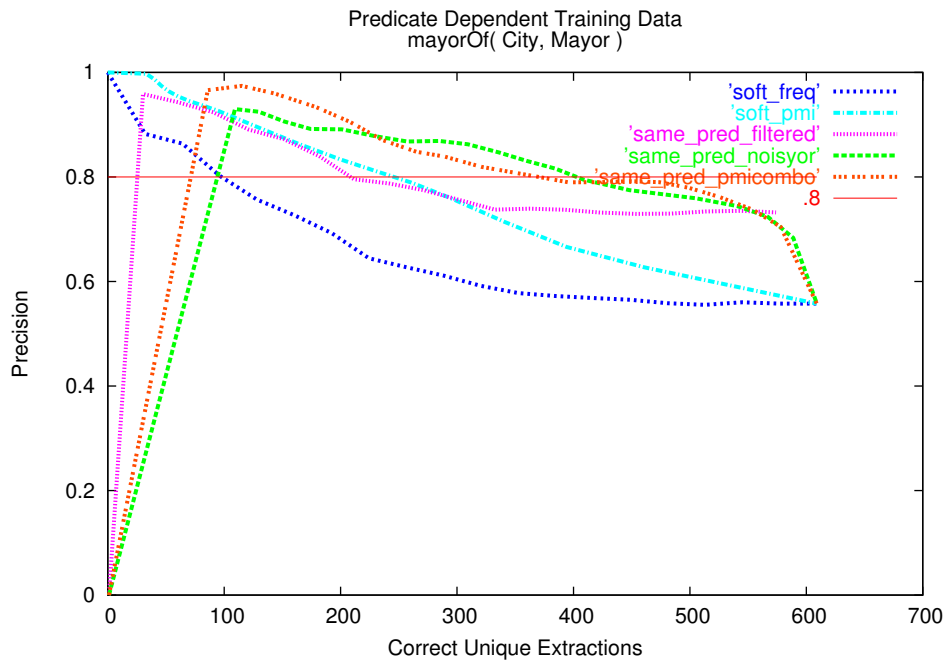
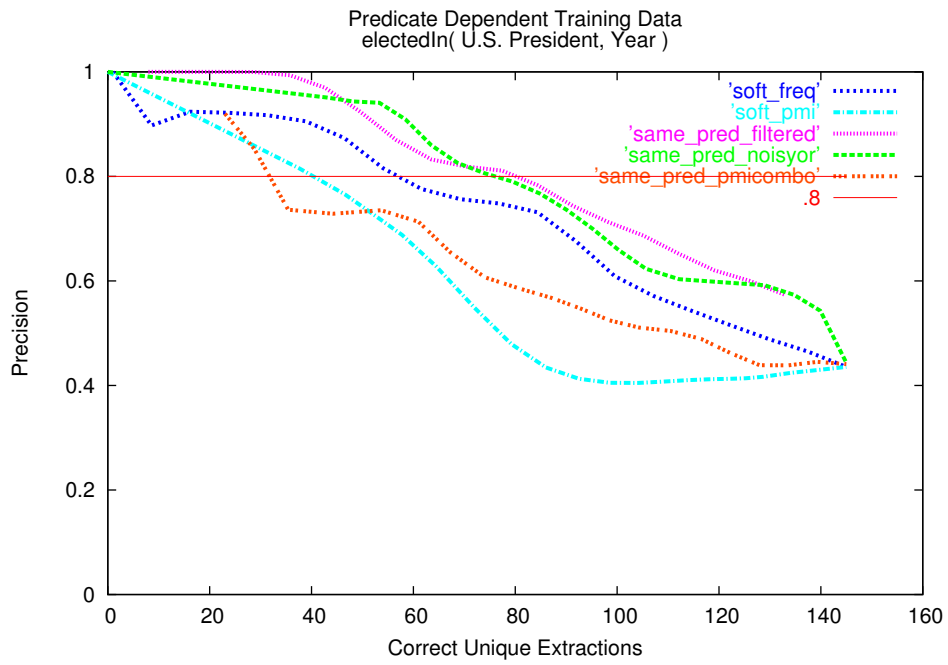


Figure 7: Results for training and testing using extractions for same predicate. (continued for next 2 pages)

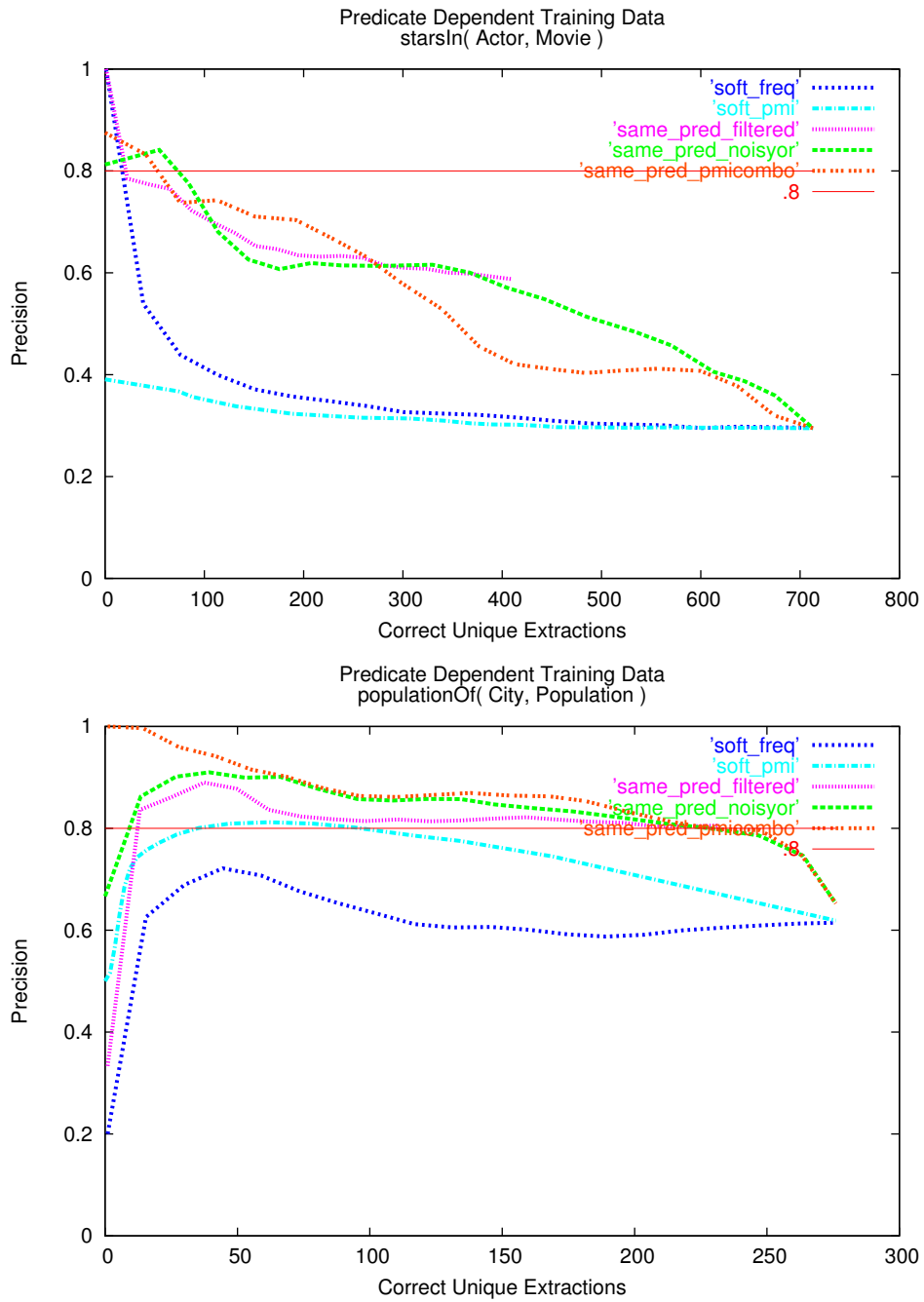


Figure 7: Results for training and testing using extractions for same predicate. (continued)

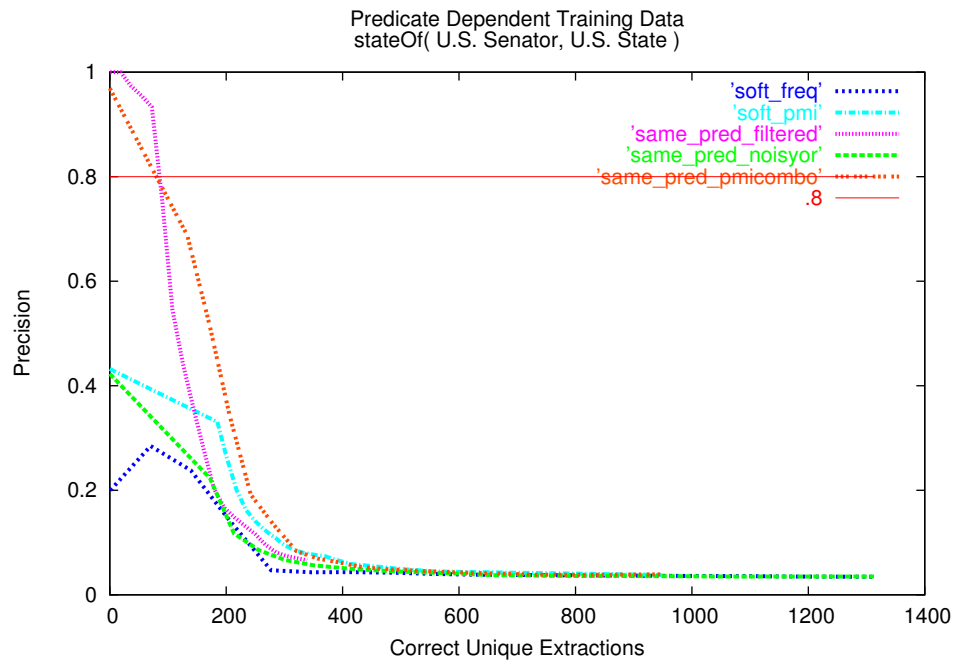


Figure 7: Results for training and testing using extractions for same predicate. (continued)

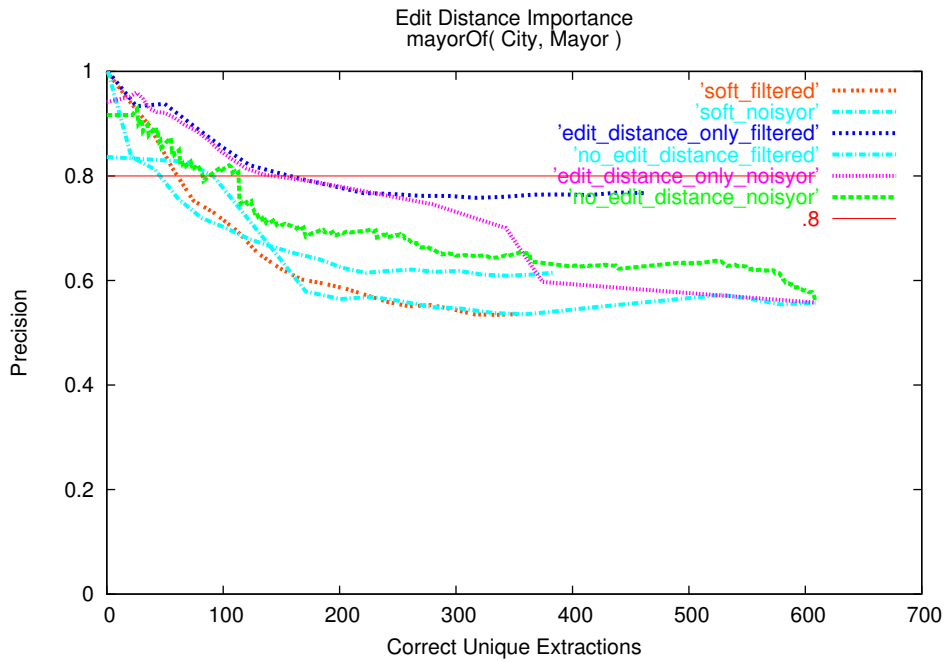
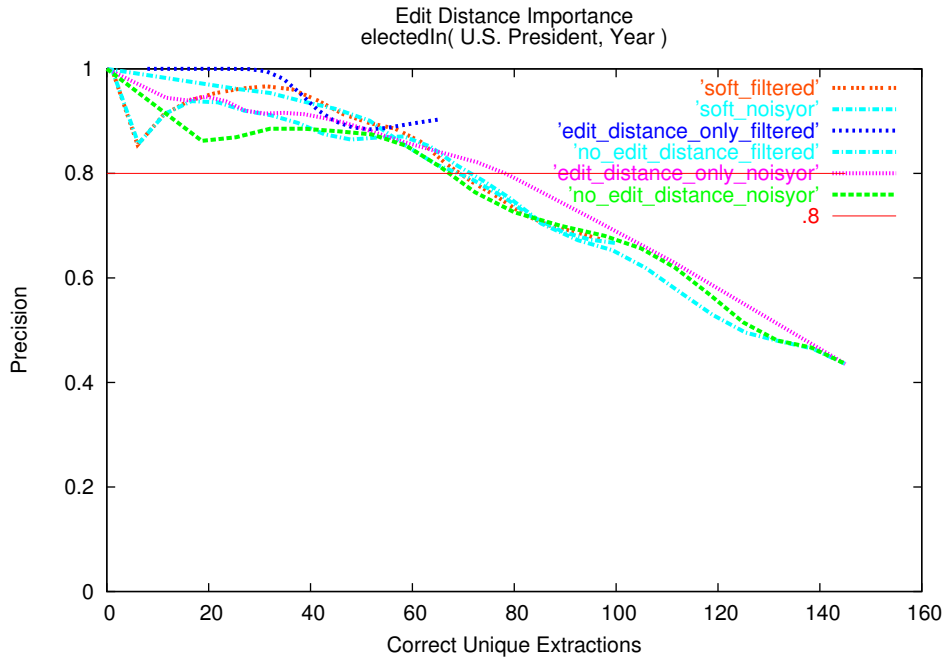


Figure 8: The Importance of the Edit Distance Feature. (continued for next 2 pages)

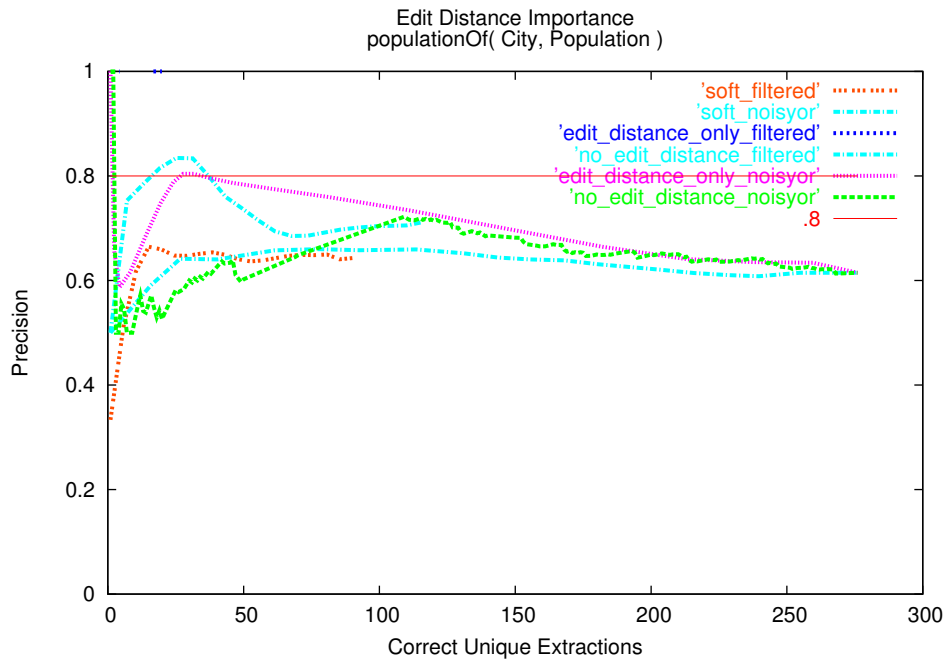
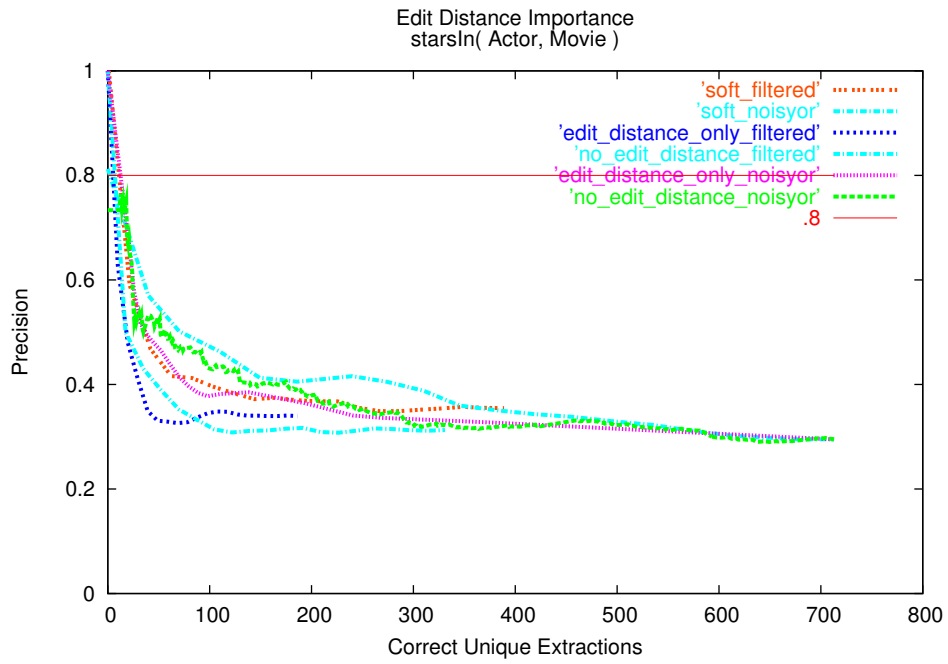


Figure 8: The Importance of the Edit Distance Feature. (continued)

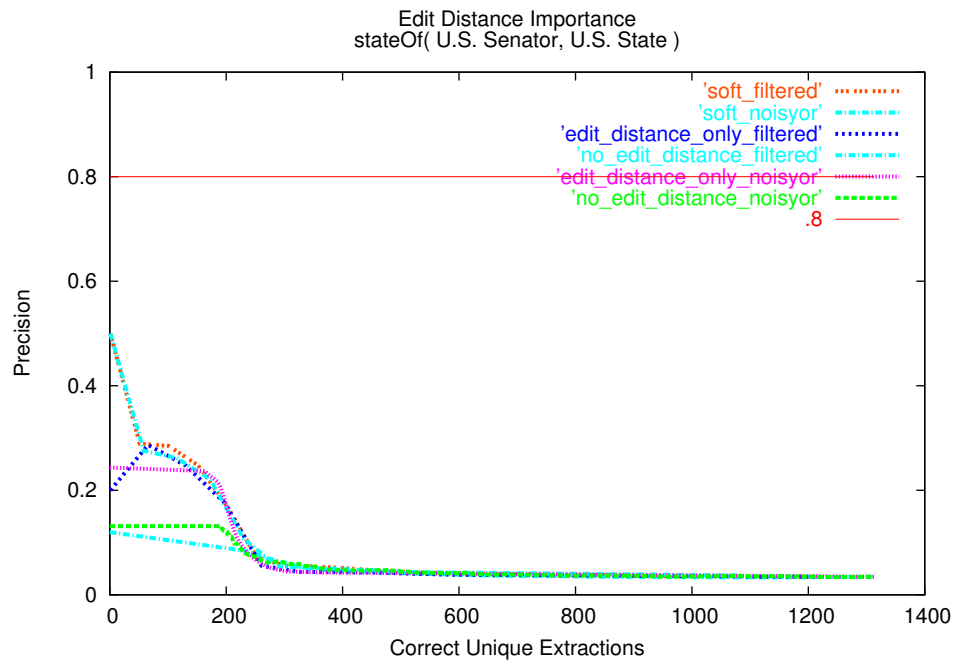


Figure 8: The Importance of the Edit Distance Feature. (continued)

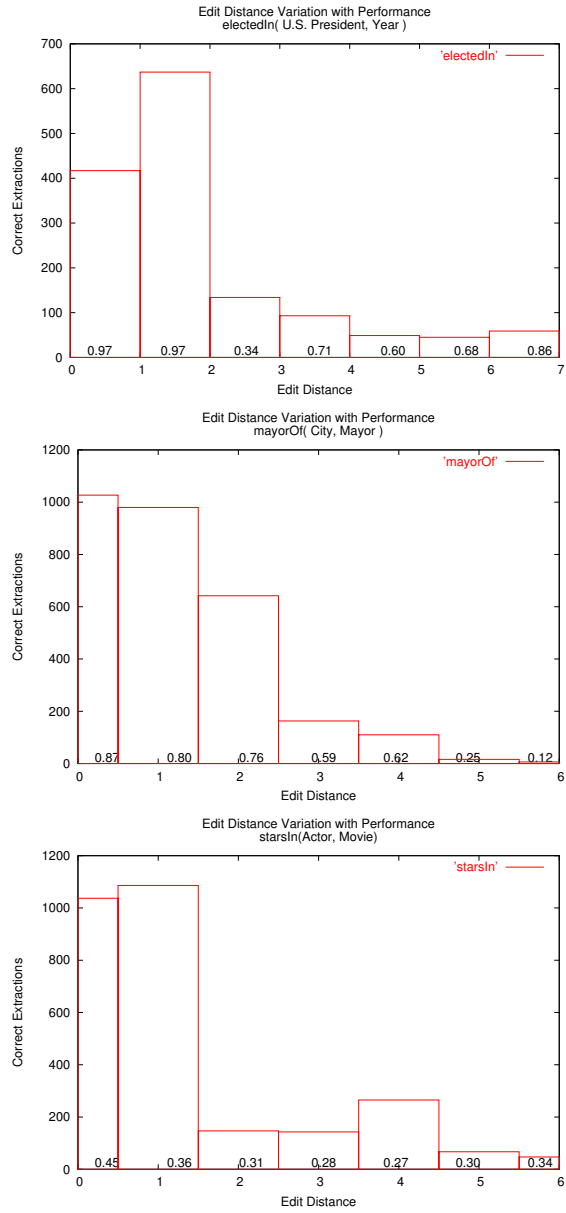


Figure 9: Rule Recall and Precision Changes With Edit Distance. The graphs above show the association between recall and edit distance. Each bar is labelled at its base with the raw precision of rules with the indicated edit distance.

References

- [1] Eugene Agichtein, Luis Gravano, Jeff Pavel, Viktoriya Sokolova, and Aleksandr Voskoboynik. Snowball: A prototype system for extracting relations from large text collections. In *SIGMOD Conference*, 2001.
- [2] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM (1)*, pages 126–134, 1999.
- [3] M.E. Califf and R.J. Mooney. Relational Learning of Pattern-Match Rules for Information Extraction. In *Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, pages 6–11, Menlo Park, CA, 1998. AAAI Press.
- [4] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. pages 22 – 29, 1990.
- [5] A. Culotta and A. McCallum. Confidence estimation for information extraction. In *HLT-NAACL*, 2004.
- [6] Oren Etzioni Douglas Downey and Stephen Soderland. A probabilistic model of redundancy in information extraction. In *To Appear: Nineteenth International Joint Conference on Artificial Intelligence*, 2005.
- [7] D. Downey, O. Etzioni, S. Soderland, and D.S. Weld. Learning Text Patterns for Web Information Extraction and Assessment. In *AAAI-04 Workshop on Adaptive Text Extraction and Mining*, pages 50–55, 2004.
- [8] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. Web-Scale Information Extraction in KnowItAll. In *WWW*, pages 100–110, New York City, New York, 2004.
- [9] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148 – 156, 1996.
- [10] Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 2004.
- [11] V. I. Levenshtein. Binary code capable of correcting spurious insertions and deletions of ones. In *Proceedings of Russian Problemy Peredachi Informatsii 1*, pages 12–25, 1965.
- [12] J.R. Quinlan. C4.5: Programs for machine learning. 1993.
- [13] E. Riloff and J. Wiebe. Learning extraction patterns for subjective expressions. In *EMNLP*, pages 105–112, 2003.
- [14] E. Riloff, J. Wiebe, and T. Wilson. Learning Subjective Nouns Using Extraction Pattern Bootstrapping. In *CoNLL*, pages 25–32s, 2003.

- [15] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press, Cambridge, MA, 2005.
- [16] S. Soderland. Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34(1-3):233–272, 1999.
- [17] Tal Shaked Stephen Soderland, Oren Etzioni and Daniel S. Weld. The use of web-based statistics to validate information extraction. In *AAAI 2004 Workshop on Adaptive Text Extraction and Mining*, 2004.
- [18] Ian H. Witten and Eibe Frank. *Data mining: Practical machine learning tools with java implementations*. San Francisco, 2000. Morgan Kaufmann.