Improved Pattern Learning for Bootstrapped Entity Extraction

Sonal Gupta **Christopher D. Manning**
Department of Computer Science
Stanford University
{sonal, manning}@cs.stanford.edu

Abstract

Bootstrapped pattern learning for entity extraction usually starts with seed entities and iteratively learns patterns and entities from unlabeled text. Patterns are scored by their ability to extract more positive entities and less negative entities. A problem is that due to the lack of labeled data, unlabeled entities are either assumed to be negative or are ignored by the existing pattern scoring measures. In this paper, we improve pattern scoring by predicting the labels of unlabeled entities. We use various unsupervised features based on contrasting domain-specific and general text, and exploiting distributional similarity and edit distances to learned entities. Our system outperforms existing pattern scoring algorithms for extracting drug-and-treatment entities from four medical forums.

1 Introduction

This paper considers the problem of building effective entity extractors for custom entity types from specialized domain corpora. We approach the problem by learning rules bootstrapped using seed sets of entities. Though entity extraction using machine learning is common in academic research, rule-based systems dominate in commercial use (Chiticariu et al., 2013), mainly because rules are effective, interpretable, and are easy to customize by non-experts to cope with errors. They also have been shown to perform better than state-of-the-art machine learning methods on some specialized domains (Nallapati and Manning, 2008; Gupta and Manning, 2014a). In addition, building supervised machine learning systems for a reasonably large domain-specific corpus would require hand-labeling sufficient data to train a model, which can be costly and time consuming. Bootstrapped machine-learned rules can make extraction easier and more efficient on such a corpus.

Seed dictionary for class 'animals': {dog}

Text:
I *own a* ***cat*** named Fluffy. I run with *my pet* ***dog***. I also nap with *my pet* ***cat***. I *own a* ***car***.

Pattern 1: my pet **X**
Extractions = positive : {dog}, unlabeled : {cat}

Pattern 2: own a **X**
Extractions = positive : {dog}, unlabeled : {car}

Figure 1: An example pattern learning system for the class 'animals' from the text. Pattern 1 and 2 are candidate patterns. Text matched with the patterns is shown in italics and the extracted entities are shown in bold.

In a bootstrapped rule-based entity learning system, seed dictionaries and/or patterns provide weak supervision to label data. The system iteratively learns new entities belonging to a specific class from unlabeled text (Riloff, 1996; Collins and Singer, 1999). Rules are typically defined by creating patterns around the entities, such as lexico-syntactic surface word patterns (Hearst, 1992) and dependency tree patterns (Yangarber et al., 2000). Patterns are scored by their ability to extract more positive entities and less negative entities. Top ranked patterns are used to extract candidate entities from text. High scoring candidate entities are added to the dictionaries and are used to generate more candidate patterns around them. In a supervised setting, the efficacy of patterns can be judged by their performance on a fully labeled dataset (Califf and Mooney, 1999; Ciravegna, 2001). In a bootstrapped system, where the data is not fully labeled, existing systems score patterns by either ignoring the un-

Proceedings of the Eighteenth Conference on Computational Language Learning, pages 98–108,
Baltimore, Maryland USA, June 26-27 2014.

labeled entities or assuming them to be negative. However, these scoring schemes cannot differentiate between patterns that extract good versus bad unlabeled entities. The problem is similar to the closed world assumption in distantly supervised information extraction systems, when all propositions missing from a knowledge base are considered false (Ritter et al., 2013; Xu et al., 2013).

Predicting labels of unlabeled entities can improve scoring patterns. Consider the example shown in Figure 1. Current pattern learning systems would score both patterns equally. However, features like distributional similarity can predict 'cat' to be closer to {dog} than 'car', and a pattern learning system can use that information to rank 'Pattern 1' higher than 'Pattern 2'.

In this paper, we work on bootstrapping entity extraction using seed sets of entities and an unlabeled text corpus. We improve the scoring of patterns for an entity class by defining a pattern's score by the number of positive entities it extracts and the ratio of number of positive entities to *expected* number of negative entities it extracts. Our main contribution is introducing the expected number of negative entities in pattern scoring – we predict probabilities of unlabeled entities belonging to the negative class. We estimate an unlabeled entity's negative class probability by averaging probabilities from various unsupervised class predictors, such as distributional similarity, string edit distances from learned entities, and TF-IDF scores. Our system performs significantly better than existing pattern scoring measures for extracting drug-and-treatment entities from four medical forums on MedHelp[1], a user health discussion website.

We release the code for the systems described in this paper at `http://nlp.stanford.edu/software/patternslearning.shtml`. We also release a visualization tool, described in Gupta and Manning (2014b), that visualizes and compares output of multiple pattern-based entity extraction systems. It can be downloaded at `http://nlp.stanford.edu/software/patternviz.shtml`.

2 Related Work

Rule based learning has been a topic of interest for many years. Patwardhan (2010) gives a good overview of the research in the field. Rule learning systems differ in how they create rules, score them, and score the entities they extract. Here, we mainly discuss the rule scoring part of the previous entity extraction research.

The pioneering work by Hearst (1992) used hand written rules to automatically generate more rules that were manually evaluated to extract hypernym-hyponym pairs from text. Other supervised systems like SRV (Freitag, 1998), SLIPPER (Cohen and Singer, 1999), $(LP)^2$ (Ciravegna, 2001), and RAPIER (Califf and Mooney, 1999) used a fully labeled corpus to either create or score rules.

Riloff (1996) used a set of seed entities to bootstrap learning of rules for entity extraction from unlabeled text. She scored a rule by a weighted conditional probability measure estimated by counting the number of positive entities among all the entities extracted by the rule. Thelen and Riloff (2002) extended the above bootstrapping algorithm for multi-class learning. Yangarber et al. (2002) and Lin et al. (2003) used a combination of accuracy and confidence of a pattern for multiclass entity learning, where the accuracy measure ignored unlabeled entities and the confidence measure treated them as negative. Gupta and Manning (2014a) used the ratio of scaled frequencies of positive entities among all extracted entities. None of the above measures predict labels of unlabeled entities to score patterns. Our system outperforms them in our experiments. Stevenson and Greenwood (2005) used Wordnet to assess patterns, which is not feasible for domains that have low coverage in Wordnet, such as medical data.

More recently, open information extraction systems have garnered attention. They focus on extracting entities and relations from the web. KnowItAll's entity extraction from the web (Downey et al., 2004; Etzioni et al., 2005) used components such as list extractors, generic and domain specific pattern learning, and subclass learning. They learned domain-specific patterns using a seed set and scored them by ignoring unlabeled entities. One of our baselines is similar to their domain-specific pattern learning component. Carlson et al. (2010) learned multiple semantic types using coupled semi-supervised training from web-scale data, which is not feasible for all datasets and entity learning tasks. They assessed patterns by their precision, assuming unla-

[1] `www.medhelp.org`

beled entities to be negative; one of our baselines is similar to their pattern assessment method.

Other open information extraction systems like ReVerb (Fader et al., 2011) and OLLIE (Mausam et al., 2012) are mainly geared towards generic, domain-independent relation extractors for web data. We tested learning an entity extractor for a given class using ReVerb. We labeled the binary and unary ReVerb extractions using the class seed entities and retrained its confidence function, with poor results. Poon and Domingos (2010) found a similar result for inducing a probabilistic ontology: an open information extraction system extracted low accuracy relational triples on a small corpus.

In this paper, we use features such as distributional similarity and edit distances from learned entities to score patterns. Similar measures have been used before but for learning entities, labeling semantic classes, or for reducing noise in seed sets (Pantel and Ravichandran, 2004; McIntosh and Curran, 2009). Measures for improving entity learning can be used alongside ours since we focus on scoring candidate patterns.

3 Approach

We use lexico-syntactic surface word patterns to extract entities from unlabeled text starting with seed dictionaries of entities for multiple classes. For ease of exposition, we present the approach below for learning entities for one class C. It can easily be generalized to multiple classes. We refer to entities belonging to C as positive and entities belonging to all other classes as negative. The bootstrapping process involves the following steps, iteratively performed until no more patterns or entities can be learned.

1. Labeling data and creating patterns: The text is labeled using the class dictionaries, starting with the seed dictionaries in the first iteration. A phrase matching a dictionary phrase is labeled with the dictionary's class. Patterns are then created using the context around the positively labeled entities to create candidate patterns for C.

2. Scoring Patterns: Candidate patterns are scored using a pattern scoring measure and the top ones are added to the list of learned patterns for C.

3. Learning entities: Learned patterns for the class are applied to the text to extract candidate entities. An entity scorer ranks the candidate entities and adds the top entities to C's dictionary.

The success of bootstrapped pattern learning methods crucially depends on the effectiveness of the pattern scorer and the entity scorer. Here we focus on improving the pattern scoring measure (Step 2 above).

3.1 Creating Patterns

Candidate patterns are created using contexts of words or their lemmas in a window of two to four words before and after a positively labeled token. Context words that are labeled with one of the classes are generalized with that class. The target term has a part-of-speech (POS) restriction, which is the POS tag of the labeled token. We create flexible patterns by ignoring the words {'a', 'an', 'the'} and quotation marks when matching patterns to the text. Some examples of the patterns are shown in Table 4.

3.2 Scoring Patterns

Judging the efficacy of patterns without using a fully labeled dataset can be challenging because of two types of failures: 1. penalizing good patterns that extract good (that is, positive) unlabeled entities, and 2. giving high scores to bad patterns that extract bad (that is, negative) unlabeled entities. Existing systems that assume unlabeled entities as negative are too conservative in scoring patterns and suffer from the first problem. Systems that ignore unlabeled entities can suffer from both the problems. In this paper, we propose to estimate the labels of unlabeled entities to more accurately score the patterns.

For a pattern r, sets P_r, N_r, and U_r denote the positive, negative, and unlabeled entities extracted by r, respectively. The pattern score, $ps(r)$ is calculated as

$$ps(r) = \frac{|P_r|}{|N_r| + \sum_{e \in U_r}(1 - score(e))} \log(|P_r|)$$

where $|.|$ denotes size of a set. The function $score(e)$ gives the probability of an entity e belonging to C. If e is a common word, $score(e)$ is 0. Otherwise, $score(e)$ is calculated as the average of five feature scores (explained below), each

of which give a score between 0 and 1. The feature scores are calculated using the seed dictionaries, learned entities for all labels, Google N-grams[2], and clustering of domain words using distributional similarity. The $\log |P_r|$ term, inspired from (Riloff, 1996), gives higher scores to patterns that extract more positive entities. Candidate patterns are ranked by $ps(r)$ and the top patterns are added to the list of learned patterns.

To calculate $score(e)$, we use features that assess unlabeled entities to be either closer to positive or negative entities in an unsupervised way. We motivate our choice of the five features below with the following insights. If the dataset consists of informally written text, many unlabeled entities are spelling mistakes and morphological variations of labeled entities. We use two edit distance based features to predict labels for these unlabeled entities. Second, some unlabeled entities are substrings of multi-word dictionary phrases but do not necessarily belong to the dictionary's class. For example, for learning drug names, the positive dictionary might contain 'asthma meds', but 'asthma' is negative and might occur in a negative dictionary as 'asthma disease'. To predict the labels of entities that are a substring of dictionary phrases, we use SemOdd, which was used in Gupta and Manning (2014a) to learn entities. Third, for a specialized domain, unlabeled entities that commonly occur in generic text are more likely to be negative. We use Google Ngrams (called GN) to get a fast, non-sparse estimate of the frequency of entities over a broad range of domains. The above features do not consider the context in which the entities occur in text. We use the fifth feature, DistSim, to exploit contextual information of the labeled entities using distributional similarity. The features are defined as:

Edit distance from positive entities (EDP): This feature gives a score of 1 if e has low edit distance to the positive entities. It is computed as $max_{p \in P_r} \mathbb{1}(\frac{editDist(p,e)}{|p|} < 0.2)$, where $\mathbb{1}(c)$ returns 1 if the condition c is true and 0 otherwise, $|p|$ is the length of p, and $editDist(p, e)$ is the Damerau-Levenshtein string edit distance between p and e.

Edit distance from negative entities (EDN): It is similar to EDP and gives a score of 1 if e has high edit distance to the negative entities. It is computed as $1 - \max_{n \in N_r} \mathbb{1}(\frac{editDist(n,e)}{|n|} < 0.2)$.

Semantic odds ratio (SemOdd): First, we calculate the ratio of frequency of the entity term in the positive entities to its frequency in the negative entities with Laplace smoothing. The ratio is then normalized using a softmax function. The feature values for the unlabeled entities extracted by all the candidate patterns are then normalized using the min-max function to scale the values between 0 and 1.[3]

Google Ngram (GN): We calculate the ratio of scaled frequency of e in the dataset to the frequency in Google Ngrams. The scaling factor is to balance the two frequencies and is computed as the ratio of total number of phrases in the dataset to the total of phrases in Google Ngrams. The feature values are normalized in the same way as SemOdd.

Distributional similarity score (DistSim): Words that occur in similar contexts, such as 'asthma' and 'depression', are clustered using distributional similarity. Unlabeled entities that get clustered with positive entities are given higher score than the ones clustered with negative entities. To score the clusters, we learn a logistic regression classifier using cluster ID as features, and use their weights as scores for all the entities in those clusters. The dataset for logistic regression is created by considering all positively labeled words as positive and sampling negative and unlabeled words as negative. The scores for entities are normalized in the same way as SemOdd and GN.

Out of feature vocabulary entities for SemOdd, GN, and DistSim are given a score of 0. We use a simple way of combining the feature values: we give equal weights to all features and average their scores. Features can be combined using a weighted average by manually tuning the weights on a development set; we leave it to the future work. Another way of weighting the features is to learn the weights using machine learning. We experimented with learning weights for

[2] `http://storage.googleapis.com/books/ngrams/books/datasetsv2.html`. Accessed January 2008.

[3] We do min-max normalization on top of the softmax normalization because the maximum and minimum value by softmax might not be close to 1 and 0, respectively. And, treating the out-of-feature-vocabulary entities same as the worst scored entities by the feature, that is giving them a score of 0, performed best on the development dataset.

the features by training a logistic regression classifier. We considered all positive words as positive and randomly sampled negative and unlabeled entities as negative to predict $score(e)$, but it performed worse compared to averaging the scores on the development dataset. Preliminary investigation suggests that since the classifier was trained on a dataset heuristically labeled using the seed dictionaries, it was too noisy for the classifier to learn accurate weights. Presumably, the classifier also suffered from the closed world assumption of treating unlabeled examples as negative.

3.3 Learning Entities

We apply the learned patterns to the text and extract candidate entities. We discard common words, negative entities, and those containing non-alphanumeric characters from the set. The rest are scored by averaging the scores of DistSim, SemOdd, EDO, and EDN features from Section 3.2 and the following features.

Pattern TF-IDF scoring (PTF): For an entity e, it is calculated as $\frac{1}{\log freq_e} \sum_{r \in R} ps(r)$, where R is the set of learned patterns that extract e and $freq_e$ is the frequency of e in the corpus. Entities that are extracted by many high weighted patterns get higher weight. To mitigate the effect of many commonly occurring entities also getting extracted by several patterns, we normalize the feature value with the log of the entity's frequency. The values are normalized in the same way as DistSim and SemOdd.

Domain N-gram TF-IDF (DN): This feature gives higher scores to entities that are more prevalent in the corpus compared to the general domain. For example, to learn entities about a specific disease from a disease-related corpus, the feature favors entities related to the disease over generic medical entities. It is calculated in the same way as GN except the frequency is computed in the n-grams of the generic domain text.

Including GN in the phrase scoring features or including DN in the pattern scoring features did not perform well on the development set in our pilot experiments.

4 Experiments

4.1 Dataset

We evaluate our system on extracting drug-and-treatment (DT) entities in sentences from four forums on the MedHelp user health discussion website: 1. Acne, 2. Adult Type II Diabetes (called Diabetes), 3. Ear Nose & Throat (called ENT), and 4. Asthma. The forums have discussion threads by users concerning health related problems and treatments. The number of sentences in each forum are: 215,623 in ENT, 39,637 in Asthma, 63,355 in Diabetes, and 65,595 in Acne. We used Asthma as the development forum for feature engineering and parameter tuning. Similar to Gupta and Manning (2014a), a DT entity is defined as a pharmaceutical drug, or any treatment or intervention mentioned that may help a symptom or a condition. It includes surgeries, lifestyle changes, alternative treatments, home remedies, and components of daily care and management of a disease, but does not include diagnostic tests and devices. More information is in the supplemental material. A few example sentences from the dataset are below.

> I plan to start **cinnamon** and **holy basil** - known to lower glucose in many people.
>
> She gave me **albuteral** and **symbicort** (plus some hayfever **meds** and asked me to use the peak flow meter.
>
> My sinus infections were treated electrically, with **high voltage million volt electricity**, which solved the problem, but the **treatment** is not FDA approved and generally unavailable, except under experimental **treatment** protocols.

In these sentences, 'cinanmon', 'holy basil', 'albuteral', 'symbicort', 'meds', 'high voltage million volt electricity', and 'treatment' are DT entities.

We used entities from the following classes as negative: symptoms and conditions (SC), medical specialists, body parts, and common temporal nouns to remove dates and dosage information. We used the DT and SC seed dictionaries from Gupta and Manning (2014a).[4] The lists of

[4]The DT seed dictionary (36,091 phrases) and SC seed dictionary (97,211 phrases) were automatically constructed from various sources on the Internet and expanded using the OAC Consumer Health Vocabulary (`http://www.consumerhealthvocab.org`), which maps medical jargon to everyday phrases and their variants. Both dictionaries are large because they contain many variants of entities. For each system, the SC dictionary was further expanded by running the system with the SC class as positive (considering DT

body parts and temporal nouns were obtained from Wordnet (Fellbaum, 1998). The common words list was created using most common words on the web and Twitter.[5]

For evaluation, the first author hand labeled the learned entities pooled from all systems. A word was evaluated by querying the word and the forum name on Google and manually inspecting the results. More details on the labeling guidelines are in the Supplement section. Inter annotator agreement between the annotator and another researcher was computed on 200 randomly sampled learned entities from each of the Asthma and ENT forum. The agreement for the entities from the Asthma forum was 96% and from the ENT forum was 92.46%. The Cohen's kappa scores were 0.91 and 0.83, respectively. Most disagreements were on food items like 'yogurt', which are hard to label. Note that we use the hand labeled entities only as a test set for evaluation.

4.2 Baselines

As in Section 3, the sets P_r, N_r, and U_r are defined as the positive, negative, and unlabeled entities extracted by a pattern r, respectively. The set A_r is defined as union of all the three sets. We compare our system with the following pattern scoring algorithms. Candidate entities are scored in the same way as described in Section 3.3. It is important to note that previous works also differ in how they create patterns, apply patterns, and score entities. Since we focus on only the pattern scoring aspect, we run experiments that differ in only that component.

PNOdd: Defined as $|P_r|/|N_r|$, this measure ignores unlabeled entities and is similar to the domain specific pattern learning component of Etzioni et al. (2005) since all patterns with $|P_r| < 2$ were discarded (more details in the next section).

PUNOdd: Defined as $|P_r|/(|U_r| + |N_r|)$, this measure treats unlabeled entities as negative entities.

RlogF: Measure used by Riloff (1996) and Thelen and Riloff (2002), and calculated as $R_r \log |P_r|$, where R_r was defined as $|P_r|/|A_r|$ (labeled RlogF-PUN). It assumed unlabeled entities as negative entities. We also compare with a variant that ignores the unlabeled entities, that is by defining R_r as $|P_r|/(|P_r + |N_r|)$ (labeled RlogF-PN).

Yangarber02: This measure from Yangarber et al. (2002) calculated two scores, $acc_r = |P_r|/|N_r|$ and $conf_r = (|P_r|/|A_r|) \log |P_r|$. Patterns with acc_r less than a threshold were discarded and the rest were ranked using $conf_r$. We empirically determined that a threshold of 0.8 performed best on the development forum.

Lin03: A measure proposed in Lin et al. (2003), it was similar to Yangarber02, except $conf_r$ was defined as $\log |P_r|(|P_r| - |N_r|)/|A_r|$. In essence, it discards a pattern if it extracts more negative entities than positive entities.

SqrtRatioAll: This pattern scoring method was used in Gupta and Manning (2014a) and defined as $\sum_{k \in P_r} \sqrt{freq_k} / \sum_{j \in A_r} \sqrt{freq_j}$, where $freq_i$ is the number of times entity i is extracted by r. Sublinear scaling of the term-frequency prevents high frequency words from overshadowing the contribution of low frequency words.

4.3 Experimental Setup

We used the same experimental setup for our system and the baselines. When matching phrases from a seed dictionary to text, a phrase is labeled with the dictionary's class if the sequence of phrase words or their lemmas match with the sequence of words of a dictionary phrase. Since our corpora are from online discussion forums, they have many spelling mistakes and morphological variations of entities. To deal with the variations, we do fuzzy matching of words – if two words are one edit distance away and are more than 6 characters long, then they are considered a match.

We used Stanford TokensRegex (Chang and Manning, 2014) to create and apply surface word patterns to text, and used the Stanford Part-of-Speech (POS) tagger (Toutanova et al., 2003) to find POS tags of tokens and lemmatize them. When creating patterns, we discarded patterns whose left or right context was 1 or 2 stop words to avoid generating low precision patterns.[6] In each iteration, we learned a maximum 20 patterns with $ps(r) \geq \theta_r$ and maximum 10 words with score $\geq$

and other classes as negative) and adding the top 50 words extracted by the top 300 patterns to the SC class dictionary. This helps in adding corpus specific SC words to the dictionary.

[5] We used top 10,000 words from Google N-grams and top 5,000 words from Twitter (`www.twitter.com`), accessed from May 19 to 25, 2012.

[6] Three or more stop words resulted in some good patterns like 'I am on **X**'. Our stop words list consists of punctuation marks and around 200 very common English words.

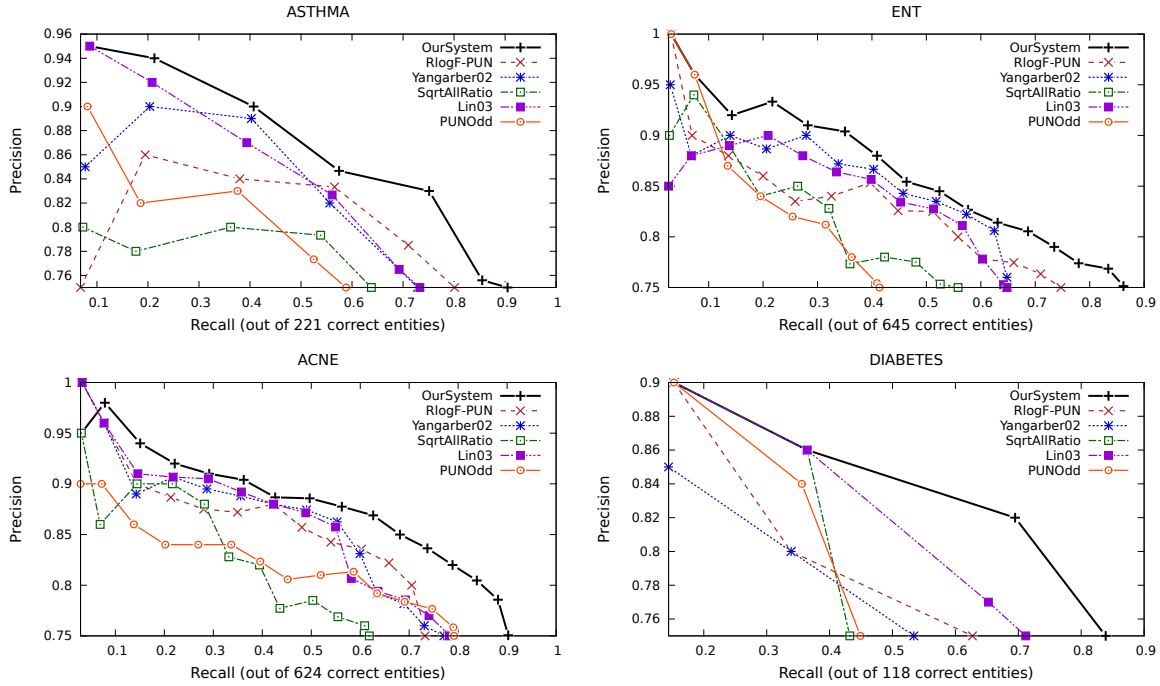

Figure 2: Precision vs. Recall curves of our system and the baselines for the four forums. Rlog-PN and PNOdd are not shown to improve clarity.

0.2. The initial value of θ_r was 1.0, which was reduced to $0.8 \times \theta_r$ whenever the system did not extract any more patterns and words. We discarded patterns that extracted less than 2 positive entities. We selected these parameters by their performance on the development forum.

For calculating the DistSim feature used for scoring patterns and entities, we clustered all of MedHelp's forum data into 1000 clusters using the Brown clustering algorithm (Brown et al., 1992; Liang, 2005).[7] For calculating the Domain Ngram feature for scoring entities, we used n-grams from all user forums in MedHelp as the domain n-grams.

We evaluate systems by their precision and recall in each iteration. Precision is defined as the fraction of correct entities among the entities extracted. We stopped learning entities for a system if the precision dropped below 75% to extract entities with reasonably high precision. Recall is defined as the fraction of correct entities among the total unique correct entities pooled from all systems while maintaining the precision $\geq 75\%$. Note that true recall is very hard to compute since our dataset is unlabeled. To compare the systems overall, we calculate the area under the precision-recall curves (AUC-PR).

System	Asthma	ENT	Diabetes	Acne
OurSystem	**68.36**	**60.71**	**67.62**	**68.01**
PNOdd	51.62	50.31	05.91	58.45
PUNOdd	42.42	30.44	36.11	58.38
RlogF-PUN	56.13	54.11	48.70	57.04
RlogF-PN	53.46	52.84	16.59	62.35
SqrtRatioAll	41.49	40.44	35.47	46.46
Yangarber02	53.76	48.46	41.45	59.85
Lin03	54.58	47.98	56.15	60.79

Table 1: Area under Precision-Recall curves of the systems.

4.4 Results

Figure 2 plots the precision and recall of systems.[8] Table 1 shows AUC-PR scores for all systems. RlogF-PN and PNOdd have low value for Diabetes because they learned generic patterns in initial iteration, which led them to learn incorrect entities. Overall our system performed significantly better than existing systems. All systems extract more entities for Acne and ENT because different drugs and treatments are more prevalent in these forums. Diabetes and Asthma have more interventions and lifestyle changes that are harder to

[7] The data consisted of around 4 million tokens. Words that occurred less than 50 times were discarded, which resulted in 50353 unique words.

[8] We do not show plots of PNOdd and RlogF-PN to improve clarity. They performed similarly to other baselines.

Feature	Asthma	ENT	Diabetes	Acne
All Features	68.36	**60.71**	67.62	68.01
EDP	**68.66**	59.07	60.03	65.15
EDN	59.39	59.21	16.75	65.96
SemOdd	67.07	58.41	60.51	65.04
GN	57.52	59.53	48.76	68.61
DistSim	64.87	59.05	**71.11**	**69.48**

Table 2: Individual feature effectiveness: Area under Precision-Recall curves when our system uses individual features during pattern scoring. Other features are still used for entity scoring.

Feature	Asthma	ENT	Diabetes	Acne
All Features	**68.36**	60.71	67.62	68.01
*minus*EDP	66.29	60.45	69.84	**69.46**
*minus*EDN	67.19	60.39	69.89	67.57
*minus*GN	65.53	60.33	66.07	67.28
*minus*SemOdd	66.66	**60.76**	**70.79**	68.25
*minus*DistSim	66.10	60.58	66.59	67.85

Table 3: Feature ablation study: Area under Precision-Recall curves when individual features are removed from our system during pattern scoring. The feature is still used for entity scoring.

extract.

To compare the effectiveness of each feature in our system, Table 2 shows the AUC-PR values when each feature was individually used for pattern scoring (other features were still used to learn entities). EDP and DistSim were strong predictors of labels of unlabeled entities because many good unlabeled entities were spelling mistakes of DT entities and occurred in similar context as them. Table 3 shows the AUC-PR values when each feature was removed from the set of features used to score patterns (the feature was still used for learning entities). Removing GN and DistSim reduced the AUC-PR scores for all forums.

Table 4 shows some examples of patterns and the entities they extracted along with their labels when the pattern was learned. We learned the first pattern because 'pinacillin' has low edit distance from the positive entity 'penicillin'. Similarly, we scored the second pattern higher than the baseline because 'desoidne' is a typo of the positive entity 'desonide'. Note that the seed dictionaries are noisy – the entity 'metro', part of the positive entity 'metrogel', was falsely considered a negative entity because it was in the common web words list. Our system learned the third pattern for two reasons: 'inhaler', 'inhalers', and 'hfa' occurred frequently as sub-phrases in the DT dictionary, and they were clustered with positive entities by distributional similarity. Since RlogF-PUN does not distinguish between unlabeled and negative entities, it is does not learn the pattern. Table 5 shows top 10 patterns learned for the ENT forum by our system and RlogF-PUN, the best performing baseline for the forum. Our system preferred to learn patterns with longer contexts, which are usually higher precision, first.

Our System	RlogF-PUN
low dose of **X***	mg of **X**
mg of **X**	treat with **X**
X 10 mg	take *DT* and **X**
she prescribe **X**	be take **X**
X 500 mg	she prescribe **X**
be take *DT* and **X***	put on **X**
ent put I on **X***	stop take **X**
DT (like X:NN	i be prescribe **X**
like *DT* and **X**	have be take **X**
then prescribe **X***	tell I to take **X**

Table 5: Top 10 (simplified) patterns learned by our system and RlogF-PUN from the ENT forum. An asterisk denotes that the pattern was never learned by the other system. **X** is the target word.

5 Discussion and Conclusion

Our system extracted entities with higher precision and recall than other existing systems. However, learning entities from an informal text corpus that is partially labeled from seed entities presents some challenges. Our system made mistakes primarily due to three reasons. One, it sometimes extracted typos of negative entities that were not easily predictable by the edit distance measures, such as 'knowwhere'. Second, patterns that extracted many good but some bad unlabeled entities got high scores because of the good unlabeled entities. However, the bad unlabeled entities extracted by the highly weighted patterns were scored high by the PTF feature during the entity scoring phase, leading to extraction of the bad entities. Better features to predict negative entities and robust text normalization would help mitigate both the problems. Third, we used automatically constructed seed dictionaries that were not dataset specific, which led to incorrectly labeling of some entities (for example, 'metro' as negative in Table 4). Reducing noise in the dictionaries would increase precision and recall.

In this paper, the features are weighted equally

Forum	Pattern	Positive entities	Negative	Unlabeled	Our System	Baseline
ENT	he give I more **X**	antibiotics, steroid, antibiotic		pinacillin	68	NA (RlogF-PUN)
Acne	topical *DT* (**X**	prednisone, clindamycin, differin, benzoyl peroxide, tretinoin, metrogel	metro	desoidne	149	231 (RlogF-PN)
Asthma	i be put on **X**	cortisone, prednisone, asmanex, advair, augmentin, bypass, nebulizer, xolair, steroids, prilosec		inhaler, inhalers, hfa	8	NA (RlogF-PUN)

Table 4: Example patterns and the entities extracted by them, along with the rank at which the pattern was added to the list of learned patterns. NA means that the system never learned the pattern. Baseline refers to the best performing baseline system on the forum. The patterns have been simplified to show just the sequence of lemmas. **X** refers to the target entity; all of them in these examples had noun POS restriction. Terms that have already been identified as the positive class were generalized to their class DT.

by taking the average of the feature scores. One area of future work is to learn weights using more sophisticated techniques; in pilot experiments, learning a logistic regression classifier on heuristically labeled data did not work well for either pattern scoring or entity scoring.

One limitation of our system and evaluation is that we learned single word entities, since calculating some features for multi-word phrases is not straightforward. For example, word clusters using distributional similarity were constructed for single words. Our future work includes expanding the features to evaluate multi-word phrases. Another avenue for future work is to use our pattern scoring method for learning other kinds of rules, such as dependency patterns, and in different kinds of systems, such as hybrid entity learning systems (Etzioni et al., 2005; Carlson et al., 2010). In addition, we did not explicitly address the problem of semantic drift (Curran et al., 2007) in this paper. In theory, learning better patterns would help lessen the problem; we plan to investigate this further.

In conclusion, we show that predicting the labels of unlabeled entities in the pattern scorer of a bootstrapped entity extraction system significantly improves precision and recall of learned entities. Our experiments demonstrate the importance of having models that contrast domain-specific and general domain text, and the usefulness of features that allow spelling variations when dealing with informal texts. Our pattern scorer outperforms existing pattern scoring methods for learning drug-and-treatment entities from four medical web forums.

Acknowledgments

We thank Diana MacLean for labeling the test data for calculating the inter-annotator agreement. We are also grateful to Gabor Angeli, Angel Chang, Manolis Savva, and the anonymous reviewers for their useful feedback. We thank MedHelp for sharing their anonymized data with us.

References

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Mary Elaine Califf and Raymond J. Mooney. 1999. Relational learning of pattern-match rules for information extraction. In *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference*, AAAI-IAAI '99, pages 328–334.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, Jr., and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 101–110.

Angel X. Chang and Christopher D. Manning. 2014. TokensRegex: Defining cascaded regular expressions over tokens. Technical Report CSTR 2014-02, Department of Computer Science, Stanford University.

Laura Chiticariu, Yunyao Li, and Frederick R. Reiss. 2013. Rule-based information extraction is dead! Long live rule-based information extraction systems! In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '13, pages 827–832.

Fabio Ciravegna. 2001. Adaptive information extraction from text by rule induction and generalisation. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, IJCAI'01, pages 1251–1256.

William W. Cohen and Yoram Singer. 1999. A simple, fast, and effective rule learner. In *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference*, pages 335–342.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110.

J. R. Curran, T. Murphy, and B. Scholz. 2007. Minimising semantic drift with mutual exclusion bootstrapping. *Proceedings of the Conference of the Pacific Association for Computational Linguistics*, pages 172–180.

D. Downey, O. Etzioni, S. Soderland, and D. S. Weld. 2004. Learning Text Patterns for Web Information Extraction and Assessment. In *Proceedings of AAAI 2004 Workshop on Adaptive Text Extraction and Mining*, ATEM '04.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91 – 134.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Dayne Freitag. 1998. Toward general-purpose learning for information extraction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, COLING-ACL '98, pages 404–408.

Sonal Gupta and Christopher D. Manning. 2014a. Induced lexico-syntactic patterns improve information extraction from online medical forums. *Under Submission*.

Sonal Gupta and Christopher D. Manning. 2014b. Spied: Stanford pattern-based information extraction and diagnostics. In *Proceedings of the ACL 2014 Workshop on Interactive Language Learning, Visualization, and Interfaces (ACL-ILLVI)*.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational linguistics*, COLING '92, pages 539–545.

Percy Liang. 2005. Semi-supervised learning for natural language. Master's thesis, MIT EECS.

Winston Lin, Roman Yangarber, and Ralph Grishman. 2003. Bootstrapped learning of semantic classes from positive and negative examples. In *Proceedings of the ICML 2003 Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 523–534.

Tara McIntosh and James R. Curran. 2009. Reducing semantic drift with bagging and distributional similarity. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, ACL-IJCNLP '09, pages 396–404.

Ramesh Nallapati and Christopher D. Manning. 2008. Legal docket-entry classification: Where machine learning stumbles. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 438–446.

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technologies*, HLT-NAACL '04, pages 321–328.

S. Patwardhan. 2010. *Widening the Field of View of Information Extraction through Sentential Event Recognition*. Ph.D. thesis, University of Utah, May.

Hoifung Poon and Pedro Domingos. 2010. Unsupervised ontology induction from text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 296–305.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence*, AAAI'96, pages 1044–1049.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction. *Transactions of the Association for Computational Linguistics*, 1:367–378.

Mark Stevenson and Mark A. Greenwood. 2005. A semantic approach to IE pattern induction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 379–386.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '02, pages 214–221.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, HLT-NAACL '03, pages 173–180.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 665–670.

Roman Yangarber, Ralph Grishman, and Pasi Tapanainen. 2000. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th International Conference on Computational Linguistics*, COLING '00, pages 940–946.

Roman Yangarber, Winston Lin, and Ralph Grishman. 2002. Unsupervised learning of generalized names. In *Proceedings of the 19th International Conference on Computational Linguistics*, COLING '02.