Helping Editors Choose Better Seed Sets for Entity Set Expansion

Vishnu Vyas
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94089
vishnuvyas@gmail.com

Patrick Pantel
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94089
me@patrickpantel.com

Eric Crestan
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94089
ecrestan@yahoo-inc.com

ABSTRACT

Sets of named entities are used heavily at commercial search engines such as Google, Yahoo and Bing. Acquiring sets of entities typically consists of combining semi-supervised expansion algorithms with manual cleaning of the resulting expanded sets. In this paper, we study the effects of different seed sets in a state-of-the-art semi-supervised expansion system and show a tremendous variation in expansion performance depending on the choice of seeds. We further show that human editors, in general, provide very bad seed sets, which perform well-below the average random seed set. We identify three factors of seed set composition, namely prototypicality, ambiguity and coverage, and we investigate their effects on expansion performance. Finally, we propose various automatic systems for improving editor-generated seed sets, which seek to remove ambiguous and other error-prone seed instances. An extensive experimental analysis shows that expansion quality, measured in R-precision, can be improved on average by a maximum of 46% by removing *the right* seeds from a seed set. Our automatic methods outperform the human editors seed sets and on average improve expansion performance by up to 34% over the original seed sets.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*concept learning*

General Terms

Algorithms, Experimentation, Measurement.

Keywords

Seed set expansion, information extraction, seed set refinement.

1. INTRODUCTION

Collections of named entities are used in many commercial and research applications, such as question answering [29] and biomedical information extraction [23]. Search engines such as Yahoo, Bing, and Google collect large sets of entities [14, 6] to better interpret queries [27], to improve query suggestions [5] and to understand query intents [11].

Manually creating and maintaing large lists of entities is expensive and laborious. In response, many automatic and semi-automatic techniques have been developed. Among them, semi-supervised techniques are most popular because they allow the users to expand specific target classes without the need for large amounts of training data. Semi-supervised methods, also called set expansion or list expansion methods, can be broadly separated into two categories, pattern based techniques [10] and distributional techniques [15].

Typical semi-supervised methods start with a small set of seed instances from a specific concept, usually obtained from an editor. Then they employ external sources of knowledge such as large corpora of text or query logs to expand the set of seeds to a larger set of candidate expansions from the same concept. For example, the seed set {*Oxygen, Mercury, Silicon*} for the concept *Elements* should return as candidates all the elements in the periodic table. However, even state-of-the-art systems inevitably produce expansions containing errors and omissions. A particular method towards cleaning expansion errors using minimal annotation effort is described in [28].

In practice, we observe that the quality of the expansion, for a given system can vary greatly based on the nature of the concept and the seed set. Different seed sets can produce widely varying results. This variation can be attributed to a variety of factors such as the ambiguity of the seeds, noise introduced by a system's particular representation of words, and the coverage of the concept's semantic space by the seeds. For example, the seed set {*Helium, Neon, Argon*} is a biased representation for the concept *Elements* because it does not completely cover the semantic space of the concept *Elements*, and it will tend to expand solely to instances of the more specific concept *Inert Gasses*.

We also observe in practice that when human editors are asked to produce seed sets, they generate widely varying sets, and generally, surprisingly, poor expansion quality.

In this paper we employ a state-of-the-art seed set expansion system [17] to study the impact of different seed sets. From six benchmark lists, we produce many random seed sets and the corresponding resulting expansions. We demonstrate large variance in their expansion performance and show that human editors choose seed sets that result in expansions of lower quality compared to an average random seed set. We then propose several algorithms for improving

CIKM'09, November 2–6, 2009, Hong Kong, China.

the seed sets given by human editors by removing seeds that tend to attract false positives. We identify three factors of seed set composition that affect the overal expansion quality and present unsupervised algorithms for improving the quality of expansions by removing seeds from the original seed sets. We empirically show, by means of an extensive evaluation, that we can improve the quality of seed sets, measured using R-precision [2], by up to 46%.

The remainder of the paper is organized as follows. In the next section we review related work and position our contribution within its landscape. In Section 3, we study the seed sets provided to us by users and exhaustively evaluate all possible removals to empirically show the wide variation in the expansion quality of the seed sets. Further, we identify three primary factors in the seed set composition which affect expansion quality. Section 4 presents algorithms for our task of seed set refinement based on the parameters of composition identified in Section 3.3 to select subsets of editorially generated seed sets. The datasets, our evaluation methodology and our experimental setup is described in Section 5 and in Section 6, we present our experimental results with an analysis of the algorithms presented in Section 4. Finally we conclude with some discussion and future work in Section 7.

2. RELATED WORK

Automatically building large lists of named entities is a common task within the information retrieval and natural language processing communities, supported by a large body of work. Various supervised, unsupervised and semi-supervised techniques have been proposed for this task. Supervised techniques work by tagging occurrences of named entities in text with coarse-grained class labels such as *People, Organization and Locations* [12, 9]. Performing finer-grained entity tagging with supervised methods requires a lot of training data. Unsupervised methods, on the other hand, rely on clustering techniques and word co-occurence patterns to extract entity sets [18, 8]. However, due to their unsupervised nature, the concepts and entity sets discovered by such methods cannot be targeted to a specific class of interest.

In practice, semi-supervised approaches are commonly used as they allow for targeting a specific class of interest, without the need for extensive training data. These methods depend on a small set of seed instances to create entity lists. They are either based on distributional approaches or use lexico-syntactic patterns to expand a seed set to a larger set of candidates. Some methods such as [22, 21] apply lexico-syntactic patterns to corpora such as web text or query logs, to expand a small set of seeds into a larger set of candidates. Other methods such as [13, 16] use the distributional hypothesis to expand seed sets.

Errors in set expansions are pervasive, even for state-of-the-art expansion systems. However, lists with high precision are important in a variety of applications. For example, most search engines use such lists in query interpretation to provide highly relevant results. Editorial annotations are commonly used, sometimes in addition to semi-supervised techniques [28] to clean entity sets generated by set expansion systems. Using user feedback is a common technique to improve system performance with minimal supervision. For example active learning methods use a group of classifiers to focus the efforts of an editor to test cases which have the maximum impact. Active learning techniques have been successfully used in a variety of natural language tasks [7, 3]. One can imagine an active learning based system which identifies bad seeds in set expansion with help from an editor.

Table 1: Seed set composition greatly impacts set expansion quality.

List	MAX	MIN	AVG	EDITOR
Elements	0.951	0.450	0.890	0.520
Roman Emperors	0.374	0.0	0.283	0.283
F1 Drivers	0.274	0.0	0.209	0.249
Countries	0.650	0.453	0.643	0.629
U.S. States	0.952	0.538	0.913	0.876
California Counties	0.333	0.0	0.147	0.143

Another approach to generating good seed sets is using completely unsupervised clustering techniques such as [20], and using the committee members generated by CBC as seed sets. Even though, the committee members are unambiguous instances from the concept and can form good seed sets, the concept that is discovered might not be what an editor wanted. Unsupervised methods, by definition are not easy to direct, and hence making it more difficult to target specific concepts.

Seed sets represent concepts. Understanding what makes a good seed set, requires understanding how people represent concepts with instances. Prototype theory [25] suggests that people represent a concept using prototypical and unambiguous instances from the concept. However, as shown in section 3.3.1, prototypical examples might actually reduce the quality of expansions. Also, there is a notion of *Basic Level* categories [26], which can be defined as partitions on the concept space that are maximally informative. We try to model this in section 4.3 and show that we can generate seed sets which have high quality expansions.

3. IMPACT OF SEED SETS

This section investigates the effect of seed set composition on expansion performance and then studies whether human editors are good at generating seed sets which lead to good expansions. Finally, we propose a set of three factors that can be attributed to the seed set composition effect.

3.1 Seed Set Composition

The performance of a set expansion system, measured as the quality of its candidate expansion can vary with the set of seeds provided to it and with the type of concept that a set expansion system is trying to expand. These variations in performance can arise from a number of factors such as the size of the seed set, the composition of the seed set and the coherence of the concept being expanded. Quality of set expansion as a function of seed size has been extensively studied in [17], where they show that gains are limited for seed sets of sizes greater than 10. In this section, we study the impact of the composition of the seed sets on the expansion performance.

To study the effect of seed set compositionality on expansion performance, we use six benchmark lists described in Section 5. We sampled 5,000 random subsets, each of size 10 from the six lists, totaling 30,000 trials. Each of these seed sets were expanded using a state-of-the-art distribu-

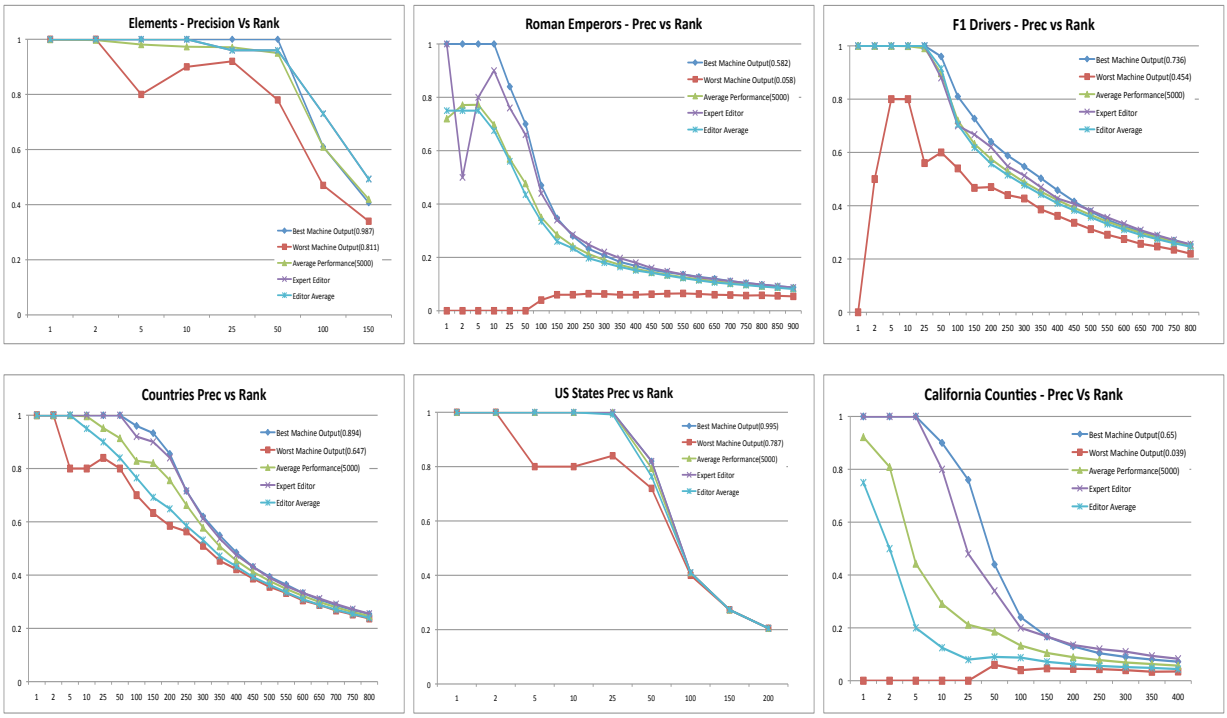

Figure 1: Precision vs. Rank curves for the six benchmark lists.

tional set expansion system [17]. Table 1 shows the expansion performance of the two random seed sets that generated the maximum and minimum performance of these seed sets, measured as the R-precision of the expansion. The table also lists the performance of an average random seed set. There is a wide variation in performance, as much as 41%, confirming that seed set composition has a significant impact on the quality of expansions.

Certain lists such as *U.S. States* and *Countries* show less variation between a random seed set and the best possible seed set than others. This suggests that some lists and the concepts they represent are easier to expand than others. On the other hand, lists such as *Roman Emperors* and *F1 Drivers* show zero R-precision with the worst possible seed set and a wide variation in expansion quality between the best performing seed set and the worst performing seed set. Thus, choosing the right seed set for these lists is critical in order to achieve high quality expansions. These conclusions are further validated by Figure 1, which shows that for the easier lists such as *U.S. States* and *Countries*, compositionality has a significant but lesser effect than that for sensitive lists such as *Roman Emperors* and *F1 Drivers* that show a tremendous variation in quality of expansions of up to 25% within the top-200 ranks.

3.2 Do humans generate good seed sets?

To see how non-expert editors (humans) compare with the performance of randomly chosen seed sets, we asked four editors to provide ten seed instances from the concepts represented by the six lists introduced in the last section. None of these editors have had any previous experience with set expansion systems. The scores were averaged over all editors and their performance is shown in Figure 1. We observe that the performance of the average editor is worse than the expansion performance of a randomly chosen set of seeds of the same size. Seed sets given by editors are not good for obtaining high quality expansions. A task in this paper is to understand why.

We also asked an expert editor, who has prior experience with set expansion systems, to provide us with 10 seeds for the same six lists. The performance of the expert editor's seed sets is also shown in Figure 1. Stunningly, an expert editor's seed sets can perform as good as the best performing random seed set. This observation is true across the six lists, where the expert editor's seed sets show consistently higher performance over the average random seed set and the average of non-expert editors. Table 1 shows the R-precision averaged over the five editors is again below the average random seed set.

To uncover the reasons why non-expert editors generate seed sets that result in poor expansion performance compared to seed sets generated by an expert editor, and to account for the variation in expansion performance with different seed sets, we studied the composition of the seed sets and identified three important factors which affect the expansion quality. The next section describes these factors in turn.

3.3 Factors in Seed Set Composition

The wide variation in the quality of set expansion due to seed composition can be attributed to three primary factors - *prototypicality*, *ambiguity* and *coverage*. While prototypicality and ambiguity are properties of the individual seeds, coverage is a property of the seed set. These factors, which are dependent on seed set composition, jointly affect the quality of set expansion systems.

Other factors such as the size and the bias in the corpus used for set expansion and type of the set expansion system

used also affect the quality of set expansion, but are not directly related to the compositionality of seed sets.

3.3.1 Prototypicality

Prototypes are words that are most representative of a class, for example words like *chair* and *table* can be considered prototypical for the class *Furniture*. Most human generated seed sets are composed mostly from prototypical examples, because they are the most natural examples humans use when expressing a concept [24].

Prototypicality of a seed depends on various factors, including the editors model of the world, the nature of the concept which it is drawn from and the dimensions of a concept. For example, the seeds *dog* and *cat* can be considered prototypical of the class *pets*, however, they are not prototypical of the class *animal*, to which they also belong. Completely modeling the prototypicality of a seed instances requires modeling the semantic space of a concept, which is difficult to say the least. Instead, we approximate the prototypicality of a seed as the likelihood of drawing the seed from a concept. This is a reasonable assumption because the more common and representative instances of a class are also more likely to be frequent in text.

In the case of set expansion systems, we observed that prototypical examples typically introduce a lot of noise in the expansions. Prototypical examples are very frequent in text and tend to occur with a wide variety of contexts that are not related to the specific class we might want to expand. For example, a prototypical example for the class of *golfers* might be *Tiger Woods*. However, the word *Tiger Woods* tends to be used in a other contexts, such as *Tiger Woods attended a charity fund raiser* and *highly paid sportsmen including Tiger Woods....* These contexts are not directly related to the class *golfers* and arise because *Tiger Woods* also belongs to another related class - *celebrities*, however, *Tiger Woods* is usually not considered a prototype for the class *celebrity*, resulting in the expansion containing other celebrities who are not golfers.

The problem is that prototypical seeds are also more likely to occur in contexts that are characteristic of a superordinate concept than other non-prototypical seeds, introducing words from the superordinate concept into expansions. Section 4.1 proposes a simple algorithm to improve the seed set's expansion quality by removing prototypical examples.

3.3.2 Ambiguity

The second compositional factor we identified as affecting the quality of expansions was the semantic ambiguity of seeds. Polysemy of a seed can introduce semantic ambiguity resulting in errors during expansion. For example, the seed set {*iron, mercury, carbon*} has two polysemous words - *iron* and *mercury*, both having varying degrees of polysemy. The seed *mercury* is highly polysemous belonging to two completely unrelated classes *elements* and *planets*, with both senses occurring equally likely. This introduces errors such as *mars, venus* and *jupiter*, all from the class *planets* into the expansion for the class *elements*.

Another source of ambiguity occurs when certain seeds are used more often in contexts that also occur with words that are not related to the primary sense of the seeds, such as in analogical and metaphorical usages. For example the seed, *Barack Obama* for the class *politicians* tends to bring in candidates from the class *athletes* because the seed is used analogically in contexts such as .. *Barack Obama won the race ...* and *running for the seat....*

Seeds which are semantically ambiguous can introduce errors in set expansions. Annotating seed instances with their senses can be impractical in most set expansion systems that depend on a large corpus because that would require annotating all the senses of seed instances within a corpus. However, as a first approximation to semantic ambiguity, we can use simple clustering technqiues to identify ambiguous seed instances. Ambiguous seeds tend to occur in contexts that are associated with words not belonging to the concept being expanded. The more ambiguous a seed instance is, the lesser is its similarity to any particular sense. Hence they can be identified as the outliers, when a set of seeds are clustered based on their semantic similarities.

Section 4.2 proposes a techniuqe using clustering algorithms such as average-link clustering to identify outliers and remove ambiguous seed instances.

3.3.3 Coverage

Another compositional factor which determines the quality of set expansion for a given seed set is the coverage of the seed set. Informally, we can define the coverage of a seed set, with respect to a concept, as the amount of semantic space which the seed set shares in common with the semantic space defined by the concept.

For example, given the concept *elements*, intuitively, the seed set {*iron, nitrogen, boron, uranium*} covers more semantic space than a set like {*helium, argon, xenon*}. The latter set is more specific and actually represents a subordinate concept of elements - *inert gasses*. If we can represent the concept as a collection of all semantic properties applicable to its instances, then we see that the seeds in second set overlap with very little semantic properties from the concept and have high overlap with other seeds in the seed set. On the other hand, seeds from the first set, whose semantic properties overlap more with the concept *elements* than among themselves are able to better represent the concept by covering more of the semantic space.

Another constraint when describing the coverage of seed sets is the inherent ambiguity within the seed sets. When a seed set contains an ambigous seed, then that seed will have lesser proportion of semantic properties in common with the concept than a more specific and unambiguous instance. Following the example of the concept *elements*, a seed instance like *mercury* shares semantic properties with both the class *elements* and other completely unrelated concepts such as *planets* and *roman gods*. On the other hand, a seed instance like *lithium*, which is a monosemous element, does not share many semantic properties outside of the class elements. Thus, we can say that the seed set containing *lithium* rather than *mercury* will generally have higher coverage of the class *element*.

Section 4.3 presents an algorithm which tries to maximize the coverage of a seed set by minimizing the semantic overlap between its elements by choosing a subset with minimum information overlap.

4. SYSTEMS

We observed in section 3 that average human editors do not produce seed sets which result in good candidate expansions. We also identified three compositional factors which affect the quality of expansions. To improve the quality of

expansions for seed sets generated by a non-expert editor, we present three algorithms, each dealing with a factor of composition, that remove seeds from the seed set that can otherwise potentially introduce errors in the expansion.

These set refinement algorithms are designed to be independent of the set expansion algorithm and once the seeds have been removed from the seed set, the expansion algorithm does not have access to the removed seeds or the original seed set. As most semi-supervised algorithms use the contexts of words to represent their semantics, they encounter similar problems and these set refinement algorithms can be used independently of the set expansion system. Also, all three algorithms presented in this section are unsupervised and do not require any extra effort from an editor.

Removing seed instances from a seed set first requires determining the number of seeds to be removed. The number of seeds to be removed depends on both the seed set being refined and the nature of the concept being expanded. Concepts that are less frequent in a corpus require less seeds to be removed to get a high quality expansion, compared to a concept that is more frequent in text. Identifying the number of seeds for each concept automatically is not dealt with in the three algorithms presented, and it is considered as a tunable model parameter. Setting of this parameter is discussed in Section 6.1.1.

4.1 Prototype Removal

The first method deals with removal of prototypes from the seed set. A prototype is a common and unambiguous instance from a concept. Also, prototypical words tend to be more common in text than specific instances from the same concept. We can statistically approximate the prototypicality of a seed s as the probability of the word given the concept C from which a word is drawn. Since, we assume, all the seeds in the seed set are drawn from the same concept, we use the seed set as an approximation to the true concept C.

Formally we define prototypicality of a seed s as

$$Prot(s) = \frac{count(s)}{\Sigma_{s' \in C} count(s')} \quad (1)$$

where the $count(s)$ for each seed s is the the number of times it occurs in a corpus.

For each seed in the seed set, we calculate its prototypicality score as defined by equation 1. Then we sort all the seeds based on their prototypicality score and remove the seeds which are most prototypical of the class, i.e, seeds with high prototypicality score. This is identical to choosing the least frequent seeds from the seed set. By removing the most prototypical seeds, we expect the contexts which are ambiguous with instances from other concepts to be filtered, thus providing us a more robust representation of the concept we are trying to expand.

4.2 Clustering

The second algorithm deals with the removal of ambiguous seeds from a seed set. Ambiguous seed instances, by definition belong to more than one concept, sometimes even belonging to two or more completly unrelated concepts. Thus, they tend to be less similar to any particular concept than their non-ambiguous counterparts. We can capture this intuition using a clustering over the words in the seed set. We expect the more non-ambigous seed instances to be more similar to the intended concept, and consequently more similar to themselves.

Any clustering over the set of seeds based on their similarities results in the ambiguous seeds being outliers. This implies that members of the tightest cluster are non-ambiguous instances from the concept represented by the seed set. We can remove the ambiguous seed instances by first clustering the seed set, and then choosing the tightest cluster as our new seed set. We ignore all other seeds that are not part of the tightest cluster.

First, we represent the semantics of each seed in the seed set by using a distributional feature vector following [17]. Each feature in the feature vector is composed of contexts that the seed occurs in. We weight the components of the feature vector in relation to the seed using point-wise mutual information between the seed and the context. We use this distributional feature representation of the seeds in an average-link clustering algorithm, where the link score between two seeds is computed as the pairwise cosine similarity between their respective feature representations.

The average-link clustering algorithm is an iterative agglomerative technique, which merges two clusters at each step using the average similarity between all the seeds in the candidate clusters. This hierarchical behavior results in a dendrogram. To retrieve a clustering from the dendrogram, we cut it using a pre-determined threshold on the number of desired clusters. The setting of the threshold is discussed in Section 6.1.1. This parameter indirectly determines the number of seeds that must be removed from the seed set. The tightness of each cluster was calculated as the average cosine similarity between all the seeds in the cluster. We chose the tightest cluster as our candidate seed set and ignore all other seeds as ambiguous.

4.3 Minimum Overlap Criterion (MOC)

The third compositional factor affecting the quality of expansions is coverage of the seed set given the concept. A seed set that better represents a concept is able to produce higher quality expansions. We can model the coverage of the concept by the seed set directly using a probabilistic argument as follows.

We define a concept as a distribution over all words in the vocabulary. Any set of words, or in our case, the seed set, jointly defines a distribution over concepts. A set of seeds S is said to have high coverage of the concept C, if it can maximize the likelihood of the concept given the set of seeds. However, the problem of directly modeling the concept space or the polysemy of a set of words is intractable. To overcome this intractability, we use a non-parametric technique to identify a subset of fixed size that can best represent the concept.

We start with the hypothesis that for a fixed size, a set of seeds which can best represent a concept C is one which provides maximum information to the concept and has minimum redundancy. To motivate the hypothesis, recall the example in section 3.3.3. The seed set {*iron, nitrogen, boron, uranium*} compared to the seed set {*helium, argon, xenon*}, provides a better representation of the concept *elements* and more coverage because the first seed set is more informative to the concept *elements* than the second seed set. From information theory, we can see this as a problem of choosing the set which has the least conditional entropy for the con-

cept given the seed set. However, choosing a subset of a fixed size from the seed set by minimizing the conditional entropy, would require us to evaluate all possible subsets.

We can simplify this process by adding another observation. The semantic overlap among the elements of the first set is lower than the semantic overlap among the elements of the second set, thus making the seeds in the second set more redundant. Thus, we need to find a subset of seeds that provides maximum information to the concept while minimizing the overlap in information among themselves.

To represent the semantics of a seed we use a vector of distributional features, which are all the contexts which appear with the seed, following the approach in [17]. To represent the semantics of the concept, we start by finding the set of features which are common to all the seeds. However, in practice, seed feature vectors are sparse and the amount of features shared between all the words is very small. To overcome this, we represent the concept with the set of features which are shared between a minimum of two seeds in the seed set.

This feature representation allows us to calculate the amount of semantic overlap between any subset of seeds given the concept, as the joint information overlap between the seeds in the seed set given the concept. We follow [19] and define the joint information overlap between a set of seeds S and a concept represented by the set of features C as:

$$I(S;C) = -\sum_{f \in C} p(f) \cdot \log(\frac{p(f \wedge S)}{p(S)}) \quad (2)$$

where $p(f)$ is the probability of seeing the feature (context) in the corpus, $p(S)$ is the probability of seeing the seed set under consideration, which can be computed as $\sum_{s \in S} p(s)$. $p(f \wedge S)$ is the joint probability of seeing the seed set S and the feature f. Since, our features are contexts surrounding the seeds, they are mutually exclusive. So, we can compute the joint probability as

$$p(f \wedge S) = \sum_{s \in S} p(f, s) \quad (3)$$

Combining Equations 2 and 3, the joint information overlap is

$$I(S;C) = \sum_{f \in C} p(f) \cdot (\log(\sum_{s \in S} p(s)) - \log(\sum_{s \in S} p(f, s)) \quad (4)$$

The function $I(S;C)$ is non-zero when the set S is not empty. Also, it is a function which assigns a score to a set, such that, given two sets, one subset of the other, the score of the subset is lower. This indicates that the function is sub modular and we use a greedy algorithm, called MOC (Minimum Overlap Criterion) to find the S that maximizes $I(S;C)$.

The algorithm starts with an empty subset and adds one element at a time at each iteration, such that the newly added element, say e has maximum marginal value with respect to the current seed set S. Here, maximum marginal value can be calculated as $I(S \cup \{e\};C) - I(S;C)$. The algorithm stops once it has reached a pre-defined size for the subset. This subset is used as the seed set for expansion. Again, the stopping criterion is considered a tunable parameter and is trained on a development set as discussed in Section 6.1.1.

5. DATASETS AND BASELINE

For evaluating the algorithms presented in Section 4, we started with nine lists of named entities, a subset of the 50 gold standard lists from [17] [1]. The 50 lists were chosen, such that each list represents a single concept. They were originally selected from wikipedia's *List of* pages. We randomly sorted the list of every noun in wikipedia. Then for each noun, we checked whether it occurred in any of the *List of* pages, if so we scraped this list - upto a maximum of 50 lists. If a noun belonged to multiple lists, the authors chose the list that seemed most appropriate. Variants within the lists were merged and the lists were manually cleaned.

The nine lists selected out of the 50 were *Elements, Roman Emperors, F1 Drivers, Countries, U.S. States, CA counties, First Ladies, American Internet Companies* and *superheroes*. For the purposes of evaluation, lists extracted from wikipedia were considered complete and treated as the gold standard.

Three of the lists, *First Ladies, American Internet Companies* and *superheroes* were designated as the development set, over which the number of seeds to be removed, which is a parameter to all our algorithms, were trained. The remaining sets were used to test the expansion performance of the seed sets generated by the three methods in Section 4.

Both the clustering technique and the MOC algorithm require statistics over distributional features of each seed. The Prototype removal algorithm requires the frequency of each seed in the seed set to compute equation 1. Wikipedia served as the source corpus for all the algorithms described in Section 4.

All articles were POS-tagged using [4] and shallow parsed (phrase chunked) using a variant of [1]. For extracting the distributional features for each seed in the seed set, we used the processed corpus to extract each seed's left and right contexts, over which all required statistics were computed. To expand the set of seeds, we employed the distributional set expansion algorithm described in [17].

6. EXPERIMENTAL RESULTS

This section presents our system analysis on the task of helping editors choose better seed sets. We begin by outlining our experimental setup and analysis metrics. Then, we show evidence that our proposed methods significantly increase expansion performance on six gold standard entity types and we present an intrinsic analysis of our algorithms addressing the three seed set composition factors introduced in Section 4. Finally, we present a detailed error analysis.

6.1 Experimental Setup

To obtain an upper bound on the expansion performance of the algorithms discussed in Section 4 we performed an exhaustive analysis. First, we created trial seed sets from the original seed sets provided to us by the editors by removing all possible combinations of seeds, of sizes ranging from 1 through 9. This resulted in 1024 trial seed sets for each list in our collection resulting in a total of 9,216 trials. Each of these trials were expanded using our set expansion algorithm [17]. We used R-precision, which is the precision

[1]The gold standard is available for download at: http://www.patrickpantel.com/cgi-bin/Web/Tools/getfile.pl?type=data&id=sse-gold/wikipedia.20071218.goldsets.tgz.

Table 2: Overall R-precision analysis over six gold standard entity types.

System	Elements	Roman Emperors	F1 Drivers	Countries	U.S. States	CA Counties	Average
MAX	0.643	0.382	0.255	0.718	0.952	0.338	0.548
USER	0.481	0.160	0.163	0.514	0.857	0.073	0.374
PROTOTYPE	0.568	0.187	0.154	0.543	0.937	0.203	0.432
CLUSTER	0.491	0.300	0.219	0.616	0.892	0.18	0.449
MOC	0.620	0.286	0.224	0.626	0.952	0.305	**0.502**

at rank equal to the size of the gold standard set to evaluate the expansions. Then, the seed sets for each list and each editor which gave the highest R-precision on their respective expansions were collected for computing the maximum upper bound on performance attainable by removing seeds from the seed sets.

For evaluation purposes, each of the expanded lists was filtered through a cosine similarity cutoff of 0.01. We used simple heuristics such as lower casing and removing special characters to merge any variants that might have occurred in the expansions. The expansions were also cleared of any seeds (or their variants) that might have been brought in by the expansions to obtain a more accurate R-precision score.

6.1.1 *Training the parameters*

Two of our algorithms, prototype removals and MOC, require the number of seeds, n, to be removed as a parameter. We empirically estimated the optimum number of seeds to be removed using our development set from Section 5. First, we used the editor generated seed sets to create candidate seed sets by using the algorithms with n ranging from 1 through 9. Each of these candidate seed sets were expanded and evaluated using R-precision. The value of n chosen was the value that gave the best R-precision averaged over all editors and all lists in the development set.

For the prototype removal algorithm, the number of seeds to be removed was determined to be 3 and for the MOC algorithm the size of the final seed set was determined to be 6, resulting in a removal of 4 seeds.

The clustering algorithm requires identifying the number of clusters to determine the cut in the dendrogram. The optimum value for this parameter was again estimated empirically over the development set. The original editor-generated seed sets were clustered over all possible values of k ranging from $k = 2$ to $k = 9$. The best value of k was determined by choosing the clustering which gave us the best average R-precision over all the editors and all the lists in the development set. This value was determined to be $k = 2$.

6.2 Overall Analysis

Table 2 lists the upper bound on R-precision and the average R-precision over all editors for each list, for the baseline (the unmodified editor seed sets), the prototype removal algorithm, clustering and MOC algorithms. Row **MAX** in Table 2 lists the upper bound of R-precision when removing seeds from the editor generated seed sets. This results in a maximum average R-precision score of 0.548 over all lists and all editors. Row **USER** lists the performance of the unmodified seeds given by the editors for each list, averaged over all editors, showing there is a maximum possible improvement in R-precision of 46%. This unmodified seed set is used as a baseline against which all our algorithms are compared.

Row **PROTOTYPE** shows the performance of the prototype removal algorithm from Section 4.1. Simply removing prototypes from the editors original seed sets can improve the average quality of expansion by as much as 15%. Row **CLUSTER** shows the performance of the clustering algorithm from Section 4.2, which chooses the tightest cluster, ignoring all other seeds. Again, like prototype removal, removing ambiguous elements through clustering increases the average R-precision over editors across all six lists, giving an average improvement in R-precision of 19%. It shows that considering and removing ambiguous elements from the seed sets improves the performance of editorially generated seed sets.

Row **MOC** describes the performance of the Minimum overlap criterion method described in Section 4.3. This method shows the highest performance out of all the three methods described in section 4. The method provides an improvement of 34% over the editors original seed sets, 12% from the upper bound.

6.3 Intrinsic Analysis of Prototype Removals and Clustering

The two techniques, prototype removal and clustering, are simple and intuitive methods for improving the quality of the seed set. Table 3 shows that both methods improve in average R-precision compared with the baseline for all five editors. The clustering method improves the seed sets of the worst performing editor, *E5* by upto 36%. The expert editor *E1**'s seed set shows the highest absolute value of R-precision, of 0.534, however for an expert editor there is a small gain in R-precision of only 9%. This expert editor reported that he was careful not to give any ambiguous seed instances for his seed set.

The above observation also holds true for the prototype removal technique, which improves the average R-precision over the baseline for all our five editors, with the R-precision gain in the expert editor's seed set being only 8%. Again, this can be explained by the fact that the expert editor reported that he was careful not to provide prototypical examples. All other non-expert editors have average R-precision gains between 17.5-19%.

6.4 Intrinsic Analysis of MOC

Table 3 shows the performance of the MOC method on a per-editor basis. MOC improves the average R-precision over the six lists for all five editors. *E1** is the expert editor whose unedited seed lists resulted in performance nearly as good as the best random set (described in section 3). It shows that MOC can improve the performance of even an expert editor by as much as 22%. It is also the case that the expert editor's improved set has the maximum absolute value of 0.601, the highest R-precision achieved among all the editors, showing that when starting with good seed sets,

Table 3: R-precision analysis for each editor. E1* is our expert editor and all other editors are non-experts.

USER	Elements	Roman Emperors	F1 Drivers	Countries	U.S. States	CA Counties	Avg. R-Prec
*E1**	0.646	0.24	0.172	0.689	0.944	0.245	0.489
E2	0.499	0.136	0.189	0.365	0.840	0.001	0.338
E3	0.407	0.078	0.158	0.558	0.919	0.055	0.362
E4	0.457	0.171	0.146	0.446	0.774	0.023	0.336
E5	0.397	0.178	0.150	0.513	0.812	0.041	0.348
PROTOTYPE							
*E1**	0.677	0.215	0.163	0.693	0.949	0.482	0.529
E2	0.503	0.184	0.179	0.424	0.949	0.087	0.387
E3	0.522	0.185	0.144	0.680	0.927	0.157	0.435
E4	0.595	0.088	0.146	0.503	0.944	0.089	0.394
E5	0.546	0.264	0.142	0.419	0.920	0.203	0.415
CLUSTER							
*E1**	0.543	0.349	0.205	0.772	0.960	0.377	0.534
E2	0.569	0.243	0.258	0.484	0.880	0.033	0.411
E3	0.569	0.276	0.205	0.541	0.880	0.148	0.436
E4	0.388	0.250	0.211	0.602	0.860	0.049	0.393
E5	0.388	0.382	0.216	0.683	0.880	0.295	0.474
MOC							
*E1**	0.722	0.403	0.228	0.735	0.952	0.566	0.601
E2	0.593	0.333	0.252	0.517	0.952	0.189	0.472
E3	0.565	0.278	0.217	0.739	0.952	0.264	0.502
E4	0.657	0.062	0.238	0.601	0.952	0.170	0.446
E5	0.565	0.354	0.187	0.538	0.952	0.340	0.489

provided by an expert editor we can potentially improve upon their original seed sets to generate candidate expansions of even higher quality.

In terms of maximum gain in R-precison, *E5*, who has the lowest average R-precision of all the editors, has an improvement of close to 50%, showing that bad seed sets provided to us by an editor can be significantly improved by considering a subset which can best represent the concept the editor is trying to expand.

MOC's high performance compared to more intuitive algorithms like prototype removal and clustering is partially caused by some aspects of those algorithms that are factored into MOC, which tries to minimize semantic overlap between the seed sets by minimizing the joint information overlap. Seeds which are prototypical tend to overlap semantically with almost all seeds in a seed set, by virtue of them being prototypical examples of a class. This leads MOC to choose seeds which are not prototypical of the class.

In MOC, the concept an editor is trying to expand is represented as a set of distributional features of all the seeds in a seed set. Ambiguous words, even though have little semantic overlap with other seeds in the seed sets, by virtue of being ambiguous do not share a lot of highly informative distributional features with the concept. This results in MOC tending to choose non-ambiguous seed instances from the seed set. This assumption is verified when looking at the selection process applied to the *U.S. states* list:

> **{*California, Arizona, Nevada, New York, New Jersey, Washington, Hawaii, Texas, Montana, Indiana*}**

This seed set by editor *E3* has ambiguous elements such as *Washington* and *New York* which are both cities and states. As expected, MOC does not choose any of the ambiguous elements, instead selecting the following seeds {*Indiana, New Jersey, California, Arizona, Nevada, Montana*}.

MOC also shows drammatic performance improvement for the list *California Counties* with editor *E5*. Table 4 shows the top-20 expansions when expanded using both the user's original seed set (*Santa Clara, Contra Costa, Fresno, San Mateo, San Joaquin, Stanislaus, Shasta, Humboldt, Mariposa, Riverside*) and MOC's subset (*Stanislaus, Mariposa, Contra Costa, San Joaquin, Shasta, San Mateo*). MOC has removed several seeds which are more known as cities than counties. This avoids errors such as *Mountain View*, *Sunnyvale* and *Bakersfield*, introduced by seeds such as *Santa Clara* which is also a city. Other errors such as *Arroyo Grande* and *Santa Rosa*, which are county heads are avoided in MOC's expansion for a similar reason. Counties such as *Fresno* and *Mariposa* are ambiguous, because they are also county heads and MOC avoids them.

MOC, by considering both prototypicality and ambiguity of seed instances, discovers a subset which can best represent the concept. Subsets chosen by MOC can improve the quality of set expansion, even in the case of expert editors.

6.5 Error Analysis

All three methods show improvement in average R-precision over the baseline scores for all five editors. Also, the MOC algorithm improves the R-precision scores of each editor for each list, with the exception of the list *Roman Emperors* for editor *E4*. Upon inspection of *E4's* seed set, we identified two sources of errors. The seed set contains the seed *Romulus*, who is the first emperor of the *Roman Republic*. However, in text, the seed *Romulus* is seen with contexts relating to the founding of Rome, rather than contexts usually associated with roman emperors. Also, the seed set contains other well known emperors such as *Nero* and *Augustus*, with a broad semantic space introducing ambiguity in our concept representation.

Table 4: Set expansions for the semantic class *California Counties* using the original seeds from editor *E5* (USER) and the automatically modified seed set using the MOC algorithm.

USER	MOC
Los Angeles	Madera
San Francisco	Napa
San Luis Obispo	Sonoma
San Bernardino	Solano
Sacramento	Tulare
Madera	Alameda
Merced	Yolo
Modesto	Del Norte
San Diego	Los Angeles
Bakersfield	Merced
Visalia	Yuba
Napa	Colusa
Tulare	*Burlingame*
Sunnyvale	Plumas
Sonoma	Tuolumne
Mountain View	Tehama
Woodlake	Siskiyou
Alameda	Calaveras
Santa Rosa	San Benito
Arroyo Grande	Mendocino

As MOC tries to choose elements which have minimum semantic overlap with each other, it chooses elements that are infrequent in the corpus. Rare elements in seed sets result in a loss of recall, and consequently loss in R-precision. This is also true for prototype removal because it directly tries to pick the least frequent elements in the seed set, causing loss in R-precision. In this case, having prototypical examples in the seed set is useful, when all the seeds are infrequent or when most of the seeds in the seed set have covered a large semantic space. A way to overcome this problem would be to vary the number of seeds to be removed on a per-list basis.

The clustering algorithm which chooses the tightest cluster also improves R-precision for all editors for all lists with the exception of the list *Elements* for two editors *E4* and *E5*. Since for each list the tightest cluster is chosen, for the editor, the tightest cluster for editor *E4* contained two elements {*Gold, Silver*}, both precious metals. The other seeds in the seed set, such as *Iron, Oxygen, Carbon, Uranium, Mercury, etc ...* were conceptually different and could not form a cohesive cluster. Similar behavior can also be observed in the case of editor *E5*, where the chosen cluster was {*Calcium, Potassium*}, both conceptually different from the other elements in the cluster such as {*Nickel, Carbon, Hydrogen, etc..*}. These errors are caused primarily because simple clustering cannot completely capture the notion of semantic ambiguity.

7. CONCLUSIONS AND FUTURE WORK

In this paper we studied the impact that seed set composition has on the quality of set expansions. We showed that the composition of seed sets can significantly affect the performance of set expansion, by as much as 41%, using a large set of random trials over an extensive evaluation. We also showed that an average editor does not produce seed sets that result in high quality expansions. In many cases, their seed sets are worse than a randomly chosen seed set from the same concept. We also asked an expert editor who creates and curates large lists of entities to provide us with seed sets of the same size. Seed sets generated by this expert editor result in expansion performance nearly as good as the best possible seed set.

We identified three important factors in seed set composition - *prototypicality, amgiguity* and *coverage* and showed that considering these factors when creating seed sets leads to higher quality expansions. We proposed three algorithms, each one tackling a different factor affecting seed set composition. The first algorithm removes prototypes identified by their relative frequency in text and we show that we can improve the quality of expansion by as much as 15%. We presented a second algorithm, which removes ambiguous seeds from a seed set through clustering analysis and then choosing the tightest cluster. This algorithm improves the quality of the expanded candidates by as much as 19%. The third algorithm tackles the third factor of seed set composition - *Coverage*. We showed that coverage can be seen as a problem in finding a subset of the seed set which best represents the concept by minimizing the semantic overlap among the seeds in the subset. Semantic overlap, can be seen as a case of minimizing the information overlap between seeds and we provided a simple greedy algorithm for minimizing the joint information overlap between seeds. We discussed how this algorithm balances between the prototypicality and ambiguity of seed instances and can choose subsets which can improve the quality of seed sets by as much as 34%.

In the future we plan to investigate algorithms for seed set refinement which are integrated with the set expansion phase. Having access to the original seed set, while giving higher preference to seeds selected by the algorithms discussed here could potentially improve the performance of set expansion. We also plan to investigate why some concepts are hard to learn and hard to expand, while others are considerably simpler to learn and expand.

In conclusion, by considering the factors of seed set compositionality when generating seed sets, either manually in the form of guidelines to the editors or automatically by refining seed sets using methods discussed above, we can substantially improve the quality set expansion systems.

8. REFERENCES

[1] S. Abney and S. P. Abney. Parsing by chunks. In *Principle-Based Parsing*, pages 257–278. Kluwer Academic Publishers, 1991.

[2] J. A. Aslam and E. Yilmaz. A geometric interpretation and analysis of r-precision. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 664–671, New York, NY, USA, 2005. ACM.

[3] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation, 2001.

[4] E. Brill. Transformation based error driven learning and natural language processing : A case study in part of speech tagging. *Computational Linguistics*, 24(4):543–565, 1995.

[5] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of KDD-08*, pages 875–883, 2008.

[6] S. Chaudhuri, V. Ganti, and D. Xin. Exploiting web search to generate synonyms for entities. In *Proceedings of WWW-09*, pages 151–160, 2009.

[7] I. Dagan and S. P. Engelson. Selective sampling in natural language learning. In *IJCAI95 Workshop On New Approaches to Learning for Natural Language Processing*, 1995.

[8] D. Downey, M. Broadhead, and O. Etzioni. Locating complex named entities in web text. In *Proc. of IJCAI*, 2007.

[9] R. Florian, , R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In *Proceedings of CoNLL-2003*, pages 168–171, 2003.

[10] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (Coling 1992)*, pages 539–545, Nantes, France, August 1992.

[11] J. Hu, G. Wang, F. Lochovsky, J. tao Sun, and Z. Chen. Understanding user's query intent with Wikipedia. In *Proceedings of WWW-09*, pages 471–480, 2009.

[12] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 188–191, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[13] M. Paşca. Organizing and searching the world wide web of facts – step two: harnessing the wisdom of the crowds. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 101–110, New York, NY, USA, 2007. ACM Press.

[14] M. Paşca. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of CIKM-07*, pages 683–690, New York, NY, USA, 2007.

[15] M. Paşca. Weakly-supervised discovery of named entities using web search queries. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 683–690, New York, NY, USA, 2007. ACM.

[16] M. Paşca and B. Van Durme. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *Proceedings of ACL-08: HLT*, pages 19–27, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[17] P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu, and V. Vyas. Web scaled distributional similarity applied to entity set extraction. In *EMNLP '09*, Singapore, 2009.

[18] P. Pantel and D. Lin. Discovering word senses from text. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619, New York, NY, USA, 2002. ACM.

[19] P. Pantel and V. Vyas. A joint information model for n-best ranking. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 681–688, Manchester, UK, August 2008. Coling 2008 Organizing Committee.

[20] P. A. Pantel. *Clustering by committee*. PhD thesis, University of Alberta, Edmonton, Alta., Canada, 2003. Adviser-Lin, Dekang.

[21] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of The Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, 1999.

[22] E. Riloff and J. Shepherd. A corpus-based approach for building semantic lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124, 1997.

[23] T. C. Rindfleisch, L. Tanabe, and J. N. . Weinstein. Edgar: Extraction of drugs, genes and relations from the biomedical literature. In *Proceedings of Pacific Symposium of Biocomputing*, pages 502–513, 2000.

[24] E. Rosch. Cognitive representations of semantic categories. *Journal of Experimental Psychology: General,*, 104(3):192–233, 1975.

[25] E. Rosch. Classification of real-world objects: Origins and representation in cognition. pages 212–222, 1977.

[26] E. Rosch. Principles of categorization. pages 27–48, 1978.

[27] B. Tan and F. Peng. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of WWW-06*, pages 1400–1405, 2006.

[28] V. Vyas and P. Pantel. Semi-automatic entity set refinement. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 290–298, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[29] R. C. Wang, N. Schlaefer, W. W. Cohen, and E. Nyberg. Automatic set expansion for list question answering. In *EMNLP*, pages 947–954. ACL, 2008.