Web-Scale Distributional Similarity and Entity Set Expansion

Patrick Pantel[†], Eric Crestan[†], Arkady Borkovsky[‡], Ana-Maria Popescu[†], Vishnu Vyas[†]

[†]Yahoo! Labs
Sunnyvale, CA 94089
{ppantel,ecrestan}@yahoo-inc.com
{amp,vishnu}@yahoo-inc.com

[‡]Yandex Labs
Burlingame, CA 94010
arkady@yandex-team.ru

Abstract

Computing the pairwise semantic similarity between all words on the Web is a computationally challenging task. Parallelization and optimizations are necessary. We propose a highly scalable implementation based on distributional similarity, implemented in the MapReduce framework and deployed over a 200 billion word crawl of the Web. The pairwise similarity between 500 million terms is computed in 50 hours using 200 quad-core nodes. We apply the learned similarity matrix to the task of automatic set expansion and present a large empirical study to quantify the effect on expansion performance of corpus size, corpus quality, seed composition and seed size. We make public an experimental testbed for set expansion analysis that includes a large collection of diverse entity sets extracted from Wikipedia.

1 Introduction

Computing the semantic similarity between terms has many applications in NLP including word classification (Turney and Littman 2003), word sense disambiguation (Yuret and Yatbaz 2009), context-spelling correction (Jones and Martin 1997), fact extraction (Paşca et al. 2006), semantic role labeling (Erk 2007), and applications in IR such as query expansion (Cao et al. 2008) and textual advertising (Chang et al. 2009).

For commercial engines such as Yahoo! and Google, creating lists of named entities found on the Web is critical for query analysis, document categorization, and ad matching. Computing term similarity is typically done by comparing co-occurrence vectors between all pairs of terms (Sarmento et al. 2007). Scaling this task to the Web requires parallelization and optimizations.

In this paper, we propose a large-scale term similarity algorithm, based on distributional similarity, implemented in the MapReduce framework and deployed over a 200 billion word crawl of the Web. The resulting similarity matrix between 500 million terms is applied to the task of expanding lists of named entities (automatic set expansion). We provide a detailed empirical analysis of the discovered named entities and quantify the effect on expansion accuracy of corpus size, corpus quality, seed composition, and seed set size.

2 Related Work

Below we review relevant work in optimizing similarity computations and automatic set expansion.

2.1 Computing Term Similarities

The distributional hypothesis (Harris 1954), which links the meaning of words to their contexts, has inspired many algorithms for computing term similarities (Lund and Burgess 1996; Lin 1998; Lee 1999; Erk and Padó 2008; Agirre et al. 2009). Brute force similarity computation compares all the contexts for each pair of terms, with complexity $O(n^2m)$ where n is the number of terms and m is the number of possible contexts. More efficient strategies are of three kinds:

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 938–947,
Singapore, 6-7 August 2009.

Smoothing: Techniques such as *Latent Semantic Analysis* reduce the context space by applying truncated Singular Value Decomposition (SVD) (Deerwester et al. 1990). Computing the matrix decomposition however does not scale well to web-size term-context matrices. Other currently unscalable smoothing techniques include Probabilistic Latent Semantic Analysis (Hofmann 1999), Iterative Scaling (Ando 2000), and Latent Dirichlet Allocation (Blei et al. 2003).

Randomized Algorithms: Randomized techniques for approximating various similarity measures have been successfully applied to term similarity (Ravichandran et al. 2005; Gorman and Curran 2006). Common techniques include *Random Indexing* based on Sparse Distributed Memory (Kanerva 1993) and *Locality Sensitive Hashing* (Broder 1997).

Optimizations and Distributed Processing: Bayardo et al. (2007) present a sparse matrix optimization strategy capable of efficiently computing the similarity between terms which's similarity exceeds a given threshold. Rychlý and Kilgarriff (2007), Elsayed et al. (2008) and Agirre et al. (2009) use reverse indexing and the MapReduce framework to distribute the similarity computations across several machines. Our proposed approach combines these two strategies and efficiently computes the exact similarity (cosine, Jaccard, Dice, and Overlap) between *all* pairs.

2.2 Entity extraction and classification

Building entity lexicons is a task of great interest for which structured, semi-structured and unstructured data have all been explored (GoogleSets; Sarmento et al. 2007; Wang and Cohen 2007; Bunescu and Mooney 2004; Etzioni et al. 2005; Paşca et al. 2006). Our own work focuses on set expansion from *unstructured* Web text. Apart from the choice of a data source, state-of-the-art entity extraction methods differ in their use of *numerous*, *few* or *no* labeled examples, the *open* or *targeted* nature of the extraction as well as the types of features employed. Supervised approaches (McCallum and Li 2003, Bunescu and Mooney 2004) rely on large sets of labeled examples, perform targeted extraction and employ a variety of sentence- and corpus-level features. While very precise, these methods are typically used for coarse grained entity classes (*People, Organizations, Companies*) for which large training data sets are available. Unsupervised approaches rely on no labeled data and use either bootstrapped class-specific extraction patterns (Etzioni et al. 2005) to find new elements of a given class (for targeted extraction) or corpus-based term similarity (Pantel and Lin 2002) to find term clusters (in an open extraction framework). Finally, semi-supervised methods have shown great promise for identifying and labeling entities (Riloff and Shepherd 1997; Riloff and Jones 1999; Banko et al. 2007; Downey et al. 2007; Paşca et al. 2006; Paşca 2007a; Paşca 2007b; Paşca and Durme 2008). Starting with a set of seed entities, semi-supervised extraction methods use either class-specific patterns to populate an entity class or *distributional similarity* to find terms similar to the seed set (Paşca's work also examines the advantages of combining these approaches). Semi-supervised methods (including ours) are useful for extending finer grain entity classes, for which large unlabeled data sets are available.

2.3 Impact of corpus on system performance

Previous work has examined the effect of using large, sometimes Web-size corpora, on system performance in the case of familiar NLP tasks. Banko and Brill (2001) show that Web-scale data helps with confusion set disambiguation while Lapata and Keller (2005) find that the Web is a good source of n-gram counts for unsupervised models. Atterer and Schutze (2006) examine the influence of corpus size on combining a supervised approach with an unsupervised one for relative clause and PP-attachment. Etzioni et al. (2005) and Pantel et al. (2004) show the advantages of using large quantities of generic Web text over smaller corpora for extracting relations and named entities. Overall, corpus size and quality are both found to be important for extraction. Our paper adds to this body of work by focusing on the task of similarity-based set expansion and providing a large empirical study quantify the relative corpus effects.

2.4 Impact of seeds on extraction performance

Previous extraction systems report on the size and quality of the training data or, if semi-supervised, the size and quality of entity or pattern seed sets. Narrowing the focus to closely related work, Paşca (2007a; 2007b) and Paşca and Durme (2008) show the impact of varying the number of instances representative of a given class and the size of the attribute seed set on the precision of class attribute extraction. An example observation is that good

Table 1. Definitions for f_0, f_1, f_2, and f_3 for commonly used similarity scores.

METRIC	$f_0(x,y,z)$	$f_1(x,y)$	$f_2(\vec{x}) = f_3(\vec{x})$
Overlap	x	1	0
Jaccard*	$\frac{x}{y+z-x}$	$\min(x,y)$	$\sum_i x_i$
Dice*	$\frac{2x}{y+z}$	$x \times y$	$\sum_i x_i^2$
Cosine	$\frac{x}{y \times z}$	$x \times y$	$\sqrt{\sum_i x_i^2}$

*weighted generalization

quality class attributes can still be extracted using 20 or even 10 instances to represent an entity class. Among others, Etzioni et al. (2005) shows that a small pattern set can help bootstrap useful entity seed sets and reports on the impact of seed set noise on final performance. Unlike previous work, empirically quantifying the influence of seed set size and quality on extraction performance of random entity types is a key objective of this paper.

3 Large-Scale Similarity Model

Term semantic models normally invoke the distributional hypothesis (Harris 1985), which links the meaning of terms to their contexts. Models are built by recording the surrounding contexts for each term in a large collection of unstructured text and storing them in a term-context matrix. Methods differ in their definition of a context (e.g., *text window* or *syntactic relations*), or by a means to weigh contexts (e.g., *frequency*, *tf-idf*, *pointwise mutual information*), or ultimately in measuring the similarity between two context vectors (e.g., using *Euclidean distance*, *Cosine*, *Dice*).

In this paper, we adopt the following methodology for computing term similarity. Our various web crawls, described in Section 6.1, are POS-tagged using Brill's tagger (1995) and chunked using a variant of the Abney chunker (Abney 1991). Terms are NP chunks with some modifiers removed; their contexts (i.e., features) are defined as their rightmost and leftmost stemmed chunks. We weigh each context f using pointwise mutual information (Church and Hanks 1989). Let PMI*(w)* denote a pointwise mutual information vector, constructed for each term as follows: $PMI(w) = (pmi_{w1}, pmi_{w2}, \ldots, pmi_{wm})$, where pmi_{wf} is the pointwise mutual information between term w and feature f:

$$pmi_{wf} = \log \frac{c_{wf} \times N}{\sum_{i=1}^{n} c_{if} \times \sum_{j=1}^{m} c_{wj}}$$

where c_{wf} is the frequency of feature f occurring for term w, n is the number of unique terms and N is the total number of features for all terms.

Term similarities are computed by comparing these *pmi* context vectors using measures such as cosine, Jaccard, and Dice.

3.1 Large-Scale Implementation

Computing the similarity between terms on a large Web crawl is a non-trivial problem, with a worst case cubic running time – $O(n^2m)$ where n is the number of terms and m is the dimensionality of the feature space. Section 2.1 introduces several optimization techniques; below we propose an algorithm for large-scale term similarity computation which calculates *exact* scores for *all* pairs of terms, generalizes to several different metrics, and is scalable to a large crawl of the Web.

Our optimization strategy follows a generalized sparse-matrix multiplication approach (Sarawagi and Kirpal 2004), which is based on the well-known observation that a scalar product of two vectors depends only on the coordinates for which both vectors have non-zero values. Further, we observe that most commonly used similarity scores for feature vectors $\vec{x}$ and $\vec{y}$, such as *cosine* and *Dice*, can be decomposed into three values: one depending only on features of $\vec{x}$, another depending only on features of $\vec{y}$, and the third depending on the features shared both by $\vec{x}$ and $\vec{y}$. More formally, commonly used similarity scores $F(\vec{x},\vec{y})$ can be expressed as:

$$F(\vec{x},\vec{y}) = f_0\left(\sum_i f_1(x_i,y_i), f_2(\vec{x}), f_3(\vec{y})\right)$$

Table 1 defines f_0, f_1, f_2, and f_3 for some common similarity functions. For each of these scores, $f_2 = f_3$. In our work, we compute all of these scores, but report our results using only the *cosine* function.

Let A and B be two matrices of PMI feature vectors. Our task is to compute the similarity between all vectors in A and all vectors in B. In computing the similarity between all pairs of terms, $A = B$.

Figure 1 outlines our algorithm for computing the similarity between all elements of A and B. Efficient computation of the similarity matrix can be achieved by leveraging the fact that $F(\vec{x},\vec{y})$ is determined solely by the features shared by $\vec{x}$ and $\vec{y}$ (i.e., $f_1(0,x) = f_1(x,0) = 0$ for any x) and that most of

the feature vectors are very sparse (i.e., most possible contexts never occur for a given term). In this case, calculating $f_1(x, y)$ is only required when both feature vectors have a shared non-zero feature, significantly reducing the cost of computation. Determining which vectors share a non-zero feature can easily be achieved by first building an inverted index for the features. The computational cost of this algorithm is $\sum N_i^2$, where N_i is the number of vectors that have a non-zero i^{th} coordinate. Its worst case time complexity is $O(ncv)$ where n is the number of terms to be compared, c is the maximum number of non-zero coordinates of any vector, and v is the number of vectors that have a non-zero i^{th} coordinate where i is the coordinate which is non-zero for the most vectors. In other words, the algorithm is efficient only when the density of the coordinates is low. On our datasets, we observed near linear running time in the corpus size.

Bayardo et al. (2007) described a strategy that potentially reduces the cost even further by omitting the coordinates with the highest number of non-zero value. However, their algorithm gives a significant advantage only when we are interested in finding solely the similarity between highly similar terms. In our experiments, we compute the exact similarity between all pairs of terms.

Distributed Implementation

The pseudo-code in Figure 1 assumes that A can fit into memory, which for large A may be impossible. Also, as each element of B is processed independently, running parallel processes for non-intersecting subsets of B makes the processing faster. In this section, we outline our MapReduce implementation of Figure 1 deployed using Hadoop[1], the open-source software package implementing the MapReduce framework and distributed file system. Hadoop has been shown to scale to several thousands of machines, allowing users to write simple "map" and "reduce" code, and to seamlessly manage the sophisticated parallel execution of the code. A good primer on MapReduce programming is in (Dean and Ghemawat 2008).

Our implementation employs the MapReduce model by using the *Map* step to start $M \times N$ *Map* tasks in parallel, each caching $1/M$th part of A as an inverted index and streaming $1/N$th part of B through it. The actual inputs are read by the tasks directly from HDFS (Hadoop Distributed File System). Each part of A is processed N times, and each part of B is processed M times. M is determined by the amount of memory dedicated for the inverted index, and N should be determined by trading off the fact that as N increases, more parallelism can be obtained at the increased cost of building the same inverse index N times.

Input: Two matrices A and B of feature vectors.

```
## Build an inverted index for A (optimiza-
## tion for data sparseness)
AA = an empty hash-table
for i in (1..n):
   F2[i] = f2(A[i]) ## cache values of f2(x)
   for k in non-zero features of A[i]:
      if k not in AA: AA[k] = empty-set
      ## append <vector-id, feature-value>
      ## pairs to the set of non-zero
      ## values for feature k
      AA[k].append( (i,A[i,k]) )
## Process the elements of B
for b in B:
   F1 = {} ## the set of Ai that have non-
        zero similarity with b
   for k in non-zero features of b:
      for i in AA[k]:
         if i not in sim: sim[i] = 0
         F1[i] += f1( AA[k][i], b[k])
   F3 = f3(b)
   for i in sim:
      print i, b, f0( F1[i], F2[i], F3)
```

Output: A matrix containing the similarity between all elements in A and in B.

Figure 1. Similarity computation algorithm.

The similarity algorithm from Figure 1 is run in each task of the *Map* step of a MapReduce job. The *Reduce* step is used to group the output by b_i.

4 Application to Set Expansion

Creating lists of named entities is a critical problem at commercial engines such as Yahoo! and Google. The types of entities to be expanded are often not known a priori, leaving supervised classifiers undesirable. Additionally, list creators typically need the ability to expand sets of varying granularity. Semi-supervised approaches are predominantly adopted since they allow targeted expansions while requiring only small sets of seed entities. State-of-the-art techniques first compute term-term similarities for all available terms and then select candidates for set expansion from amongst the terms most similar to the seeds (Sarmento et al. 2007).

[1] Hadoop, http://lucene.apache.org/hadoop/

Formally, we define our expansion task as:

> **Task Definition**: Given a set of seed entities $S = \{s_1, s_2, ..., s_k\}$ of a class $C = \{s_1, s_2, ..., s_k, ..., s_n\}$ and an unlabeled textual corpus T, find all members of the class C.

For example, consider the class of *Bottled Water Brands*. Given the set of seeds S = {*Volvic*, *San Pellegrino*, *Gerolsteiner Brunnen, Bling* H^2O}, our task is to find all other members of this class, such as {*Agua Vida*, *Apenta, Culligan*, *Dasani*, *Ethos Water*, *Iceland Pure Spring Water*, *Imsdal*, …}

4.1 Set Expansion Algorithm

Our goal is not to propose a new set expansion algorithm, but instead to test the effect of using our Web-scale term similarity matrix (enabled by the algorithm proposed in Section 3) on a state-of-the-art distributional set expansion algorithm, namely (Sarmento et al. 2007).

We consider S as a set of prototypical examples of the underlying entity set. A representation for the meaning of S is computed by building a feature vector consisting of a weighted average of the features of its seed elements $s_1, s_2, ..., s_k$, a centroid. For example, given the seed elements {*Volvic*, *San Pellegrino*, *Gerolsteiner Brunnen, Bling* H^2O}, the resulting centroid consists of (details of the feature extraction protocol are in Section 6.1):

```
brand, mineral water, monitor,
lake, water, take over, …
```

Centroids are represented in the same space as terms allowing us to compute the similarity between centroids and all terms in our corpus. A scored and ranked set for expansion is ultimately generated by sorting all terms according to their similarity to the seed set centroid, and applying a cutoff on either the similarity score or on the total number of retrieved terms. In our reported experiments, we expanded over 22,000 seed sets using our Web similarity model from Section 3.

5 Evaluation Methodology

In this section, we describe our methodology for evaluating Web-scale set expansion.

5.1 Gold Standard Entity Sets

Estimating the quality of a set expansion algorithm requires a random sample from the universe of all entity sets that may ever be expanded, where a set represents some concept such as *Stage Actors*. An approximation of this universe can be extracted from the "*List of*" pages in Wikipedia[2].

Upon inspection of a random sample of the "*List of*" pages, we found that several lists were compositions or joins of concepts, for example "*List of World War II aces from Denmark*" and "*List of people who claimed to be God*". We addressed this issue by constructing a quasi-random sample as follows. We randomly sorted the list of every noun occurring in Wikipedia[2]. Then, for each noun we verified whether or not it existed in a Wikipedia list, and if so we extracted this list. If a noun belonged to multiple lists, the authors chose the list that seemed most appropriate. Although this does not generate a perfect random sample, diversity is ensured by the random selection of nouns and relevancy is ensured by the author adjudication.

The final gold standard consists of 50 sets, including: *classical pianists*, *Spanish provinces*, *Texas counties*, *male tennis players*, *first ladies*, *cocktails*, *bottled water brands*, and *Archbishops of Canterbury*. For each set, we then manually scraped every instance from Wikipedia keeping track also of the listed variants names.

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

The 50 sets consist on average of 208 instances (with a minimum of 11 and a maximum of 1,116) for a total of 10,377 instances.

5.2 Trials

In order to analyze the corpus and seed effects on performance, we created 30 copies of each of the 50 sets and randomly sorted each copy. Then, for each of the 1500 copies, we created a trial for each of the following 23 seed sizes: 1, 2, 5, 10, 20, 30, 40, …, 200. Each trial of seed size s was created by taking the first s entries in each of the 1500 random copies. For sets that contained fewer than 200 items, we only generated trials for seed sizes

[2] In this paper, extractions from Wikipedia are taken from a snapshot of the resource in December 2008.

Table 2. Corpora used to build our expansion models.

CORPORA	UNIQUE SENTENCES (MILLIONS)	TOKENS (MILLIONS)	UNIQUE WORDS (MILLIONS)
Web100	5,201	217,940	542
Web020†	1040	43,588	108
Web004†	208	8,717	22
Wikipedia[6]	30	721	34

†Estimated from *Web100* statistics.

smaller than the set size. The resulting trial dataset consists of 20,220 trials[3].

5.3 Judgments

Set expansion systems consist of an expansion algorithm (such as the one described in Section 4.1) as well as a corpus (such as Wikipedia, a news corpus, or a web crawl). For a given system, each of the 20,220 trials described in the previous section are expanded. In our work, we limited the total number of system expansions, per trial, to 1000.

Before judgment of an expanded set, we first collapse each instance that is a variant of another (determined using the variants in our gold standard) into one single instance (keeping the highest system score)[4]. Then, each expanded instance is judged as *correct* or *incorrect* automatically against the gold standard described in Section 5.1.

5.4 Analysis Metrics

Our experiments in Section 6 consist of precision vs. recall or precision vs. rank curves, where:

a) ***precision*** is defined as the percentage of correct instances in the expansion of a seed set; and

b) ***recall*** is defined as the percentage of non-seed gold standard instances retrieved by the system.

Since the gold standard sets vary significantly in size, we also provide the *R*-precision metric to normalize for set size:

c) ***R-precision*** is defined as the average precision of all trials where precision is taken at rank R = {size of trial's associated gold standard set}, thereby normalizing for set size.

For the above metrics, 95% confidence bounds are computed using the randomly generated samples described in Section 5.2.

6 Experimental Results

Our goal is to study the performance gains on set expansion using our Web-scale term similarity algorithm from Section 3. We present a large empirical study quantifying the importance of corpus and seeds on expansion accuracy.

6.1 Experimental Setup

We extracted statistics to build our model from Section 3 using four different corpora, outlined in Table 2. The *Wikipedia* corpus consists of a snapshot of the *English* articles in December 2008[5]. The *Web100* corpus consists of an extraction from a large crawl of the Web, from Yahoo!, of over 600 million English webpages. For each crawled document, we removed paragraphs containing fewer than 50 tokens (as a rough approximation of the narrative part of a webpage) and then removed all duplicate sentences. The resulting corpus consists of over 200 billion words. The *Web020* corpus is a random sample of $1/5^{th}$ of the sentences in *Web100* whereas *Web004* is a random sample of $1/25^{th}$ of *Web100*.

For each corpus, we tagged and chunked each sentence as described in Section 3. We then computed the similarity between all noun phrase chunks using the model of Section 3.1.

6.2 Quantitative Analysis

Our proposed optimization for term similarity computation produces *exact* scores (unlike randomized techniques) for *all* pairs of terms on a large Web crawl. For our largest corpus, *Web100*, we computed the pairwise similarity between over 500 million words in 50 hours using 200 four-core machines. Web004 is of similar scale to the largest reported randomized technique (Ravichandran et al. 2005). On this scale, we compute the exact similarity matrix in a little over two hours whereas Ravichandran et al. (2005) compute an approximation in 570 hours. On average they only find 73%

[3] Available for download at http://www.patrickpantel.com/cgi-bin/Web/Tools/getfile.pl?type=data&id=sse-gold/wikipedia.20071218.trials.tgz.

[4] Note also that we do not allow seed instances nor their variants to appear in an expansion set.

[5] To avoid biasing our Wikipedia corpus with the test sets, Wikipedia "*List of*" pages were omitted from our statistics as were any page linked to gold standard list members from "List of" pages.

Table 3. Corpora analysis: *R*-precision and *Precision* at various ranks. 95% confidence bounds are all below 0.005†.

CORPORA	R-PREC	PREC@25	PREC@50	PREC@100
Web100	**0.404**	**0.407**	**0.347**	**0.278**
Web020	0.356	0.377	0.319	0.250
Web004	0.264	0.353	0.298	0.239
Wikipedia	0.315	0.372	0.314	0.253

†95% confidence bounds are computed over all trials described in Section 5.2.

of the top-1000 similar terms of a random term whereas we find all of them.

For set expansion, experiments have been run on corpora as large as *Web004* and Wikipedia (Sarmento et al. 2007), a corpora 300 times smaller than our Web crawl. Below, we compare the expansion accuracy of Sarmento et al. (2007) on Wikipedia and our Web crawls.

Figure 2 illustrates the precision and recall tradeoff for our four corpora, with 95% confidence intervals computed over all 20,220 trials described in Section 4.2. Table 3 lists the resulting *R*-precision along with the system precisions at ranks 25, 50, and 100 (see Figure 2 for detailed precision analysis). Why are the *precision* scores so low? Compared with previous work that manually select entity types for expansion, such as *countries* and *companies*, our work is the first to evaluate over a large set of randomly selected entity types. On just the *countries* class, our *R*-Precision was *0.816* using *Web100*.

The following sections analyze the effects of various expansion variables: corpus size, corpus quality, seed size, and seed quality.

6.2.1 Corpus Size and Corpus Quality Effect

Not surprisingly, corpus size and quality have a significant impact on expansion performance. Figure 2 and Table 3 quantify this expectation. On our Web crawl corpora, we observe that the full 200+ billion token crawl (*Web100*) has an average *R*-precision 13% higher than 1/5th of the crawl (*Web020*) and 53% higher than 1/25th of the crawl. Figure 2 also illustrates that throughout the full precision/recall curve, *Web100* significantly outperforms *Web020*, which in turn significantly outperforms *Web004*.

The higher text quality *Wikipedia* corpus, which consists of roughly 60 times fewer tokens than

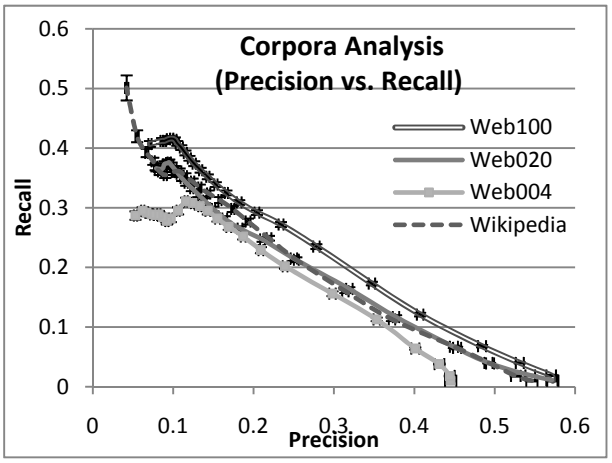

Figure 2. Corpus size and quality improve performance.

Web020, performs nearly as well as *Web020* (see Figure 2). We omitted statistics from Wikipedia "*List of*" pages in order to not bias our evaluation to the test set described in Section 5.1. Inspection of the precision vs. rank graph (omitted for lack of space) revealed that from rank 1 thru 550, *Wikipedia* had the same precision as *Web020*. From rank 550 to 1000, however, *Wikipedia*'s precision dropped off significantly compared with *Web020*, accounting for the fact that the Web corpus contains a higher recall of gold standard instances. The *R*-precision reported in Table 3 shows that this precision drop-off results in a significantly lower *R*-precision for *Wikipedia* compared with *Web020*.

6.2.2 The Effect of Seed Selection

Intuitively, some seeds are better than others. We study the impact of seed selection effect by inspecting the system performance for several randomly selected seed sets of fixed size and we find that ***seed set composition greatly affects performance***. Figure 3 illustrates the precision vs. recall tradeoff on our best performing corpus *Web100* for 30 random seed sets of size 10 for each of our 50 gold standard sets (i.e., 1500 trials were tested.) Each of the trials performed better than the average system performance (the double-lined curve lowest in Figure 3). Distinguishing between the various data series is not important, however important to notice is the very large gap between the precision/recall curves of the best and worst performing random seed sets. On average, the best performing seed sets had 42% higher precision and 39% higher recall than the worst performing seed set. Similar

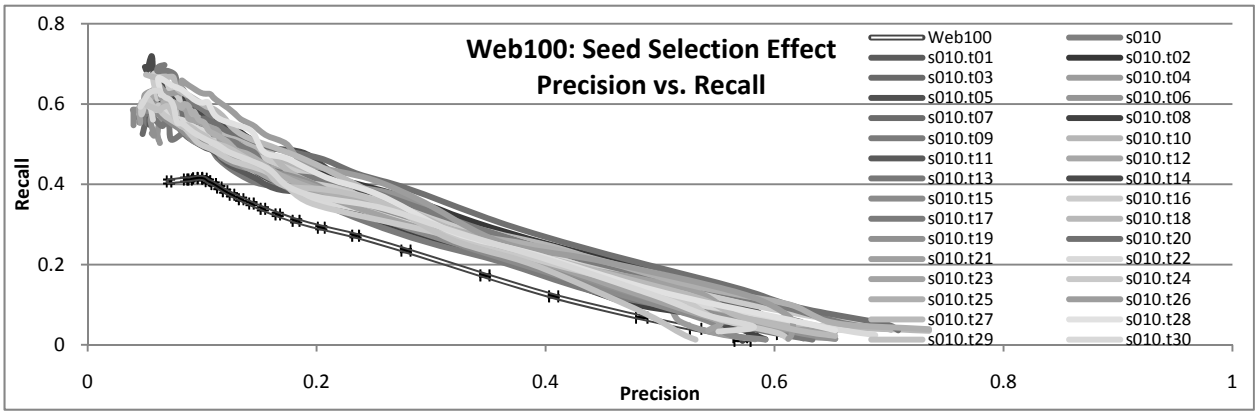

Figure 3. Seed set composition greatly affects system performance (with 30 different seed samples of size 10).

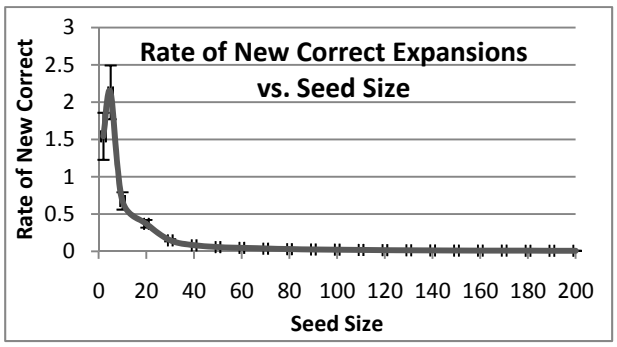

Figure 4. Few new instances are discovered with more than 5-20 seeds on *Web100* (with 95% confidence).

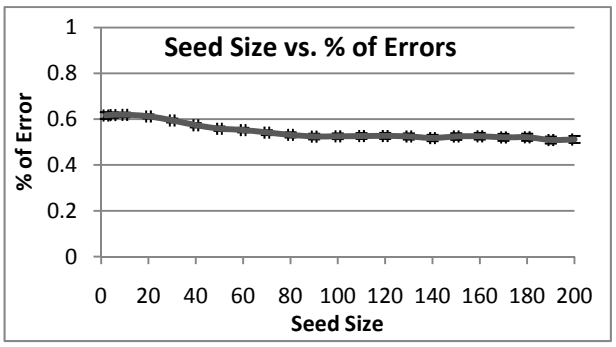

Figure 5. Percentage of errors does not increase as seed size increases on *Web100* (with 95% confidence).

curves were observed for inspected seed sets of size 5, 20, 30, and 40.

Although outside of the scope of this paper, we are currently investigating ways to automatically detect which seed elements are better than others in order to reduce the impact of seed selection effect.

6.2.3 The Effect of Seed Size

Here we aim to confirm, with a large empirical study, the anecdotal claims in (Paşca and Durme 2008) that few seeds are necessary. We found that a) very small seed sets of size 1 or 2 are not sufficient for representing the intended entity set; b) 5-20 seeds yield on average best performance; and c) surprisingly, ***increasing the seed set size beyond 20 or 30 on average does not find any new correct instances***.

We inspected the effect of seed size on *R*-precision over the four corpora. Each seed size curve is computed by averaging the system performance over the 30 random trials of all 50 sets. For each corpus, *R*-precision increased sharply from seed size 1 to 10 and the curve flattened out for seed sizes larger than 20 (figure omitted for lack of space). Error analysis on the *Web100* corpus shows that once our model has seen 10-20 seeds, the distributional similarity model seems to have enough statistics to discover as many new correct instances as it could ever find. Some entities could never be found by the distributional similarity model since they either do not occur or infrequently occur in the corpus or they occur in contexts that vary a great deal from other set elements. Figure 4 illustrates this behavior by plotting for each seed set size the rate of increase in discovery of *new* correct instances (i.e., not found in smaller seed set sizes).

We see that most gold standard instances are discovered with the first 5-10 seeds. After the 30th seed is introduced, no new correct instances are found. An important finding is that the ***error rate does not increase with increased seed set size*** (see Figure 5). This study shows that only few seeds (10-20) yield best performance and that adding more seeds beyond this does not on average affect performance in a positive or negative way.

7 Conclusion

We proposed a highly scalable term similarity algorithm, implemented in the MapReduce framework, and deployed over a 200 billion word crawl of the Web. The pairwise similarity between 500 million terms was computed in 50 hours using 200 quad-core nodes. We evaluated the impact of the large similarity matrix on a set expansion task and found that the Web similarity matrix gave a large performance boost over a state-of-the-art expansion algorithm using Wikipedia. Finally, we release to the community a testbed for experimentally analyzing automatic set expansion, which includes a large collection of nearly random entity sets extracted from Wikipedia and over 22,000 randomly sampled seed expansion trials.

References

Abney, S. Parsing by Chunks. In: Robert Berwick, Steven Abney and Carol Tenny (eds.), *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht. 1991.

Agirre, E.; Alfonseca, E.; Hall, K.; Kravalova, J.; Pasca, M.; and Soroa, A.. 2009. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In *Proceedings of NAACL HLT 09*.

Ando, R. K. 2000. Latent semantic space: Iterative scaling improves precision of interdocument similarity measurement. In *Proceedings of SIGIR-00*. pp. 216–223.

Atterer, M. and Schutze, H., 2006. The Effect of Corpus Size when Combining Supervised and Unsupervised Training for Disambiguation. In *Proceedings of ACL-06*.

Banko, M. and Brill, E. 2001. Mitigating the paucity of data problem. In *Proceedings of HLT-2001*. San Diego, CA.

Banko, M.; Cafarella, M.; Soderland, S.; Broadhead, M.; Etzioni, O. 2007. Open Information Extraction from the Web. In *Proceedings of IJCAI*.

Bayardo, R. J.; Ma, Y.; Srikant, R. 2007. Scaling Up All-Pairs Similarity Search. *In Proceedings of WWW-07*. pp. 131-140. Banff, Canada.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Brill, E. 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging. *Computational Linguistics*.

Broder, A. 1997. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*. pp. 21-29.

Bunescu, R. and Mooney, R. 2004 Collective Information Extraction with Relational Markov Networks. In *Proceedings of ACL-04,* pp. 438-445.

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

Chang, W.; Pantel, P.; Popescu, A.-M.; and Gabrilovich, E. 2009. Towards intent-driven bidterm suggestion. In *Proceedings of WWW-09 (Short Paper)*, Madrid, Spain.

Church, K. and Hanks, P. 1989. Word association norms, mutual information, and lexicography. In *Proceedings of ACL89*. pp. 76–83.

Dean, J. and Ghemawat, S. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107-113.

Deerwester, S. C.; Dumais, S. T.; Landauer, T. K.; Furnas, G. W.; and Harshman, R. A. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Downey, D.; Broadhead, M; Etzioni, O. 2007. Locating Complex Named Entities in Web Text. In *Proceedings of IJCAI-07.*

Elsayed, T.; Lin, J.; Oard, D. 2008. Pairwise Document Similarity in Large Collections with MapReduce. In *Proceedings of ACL-08: HLT, Short Papers (Companion Volume)*. pp. 265–268. Columbus, OH.

Erk, K. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of ACL-07*. pp. 216–223. Prague, Czech Republic.

Erk, K. and Padó, S. 2008. A structured vector space model for word meaning in context. In *Proceedings of EMNLP-08.* Honolulu, HI.

Etzioni, O.; Cafarella, M.; Downey. D.; Popescu, A.; Shaked, T; Soderland, S.; Weld, D.; Yates, A. 2005. Unsupervised named-entity extraction from the Web: An Experimental Study. In *Artificial Intelligence,* 165(1):91-134.

Gorman, J. and Curran, J. R. 2006. Scaling distributional similarity to large corpora. In Proceedings of ACL-06. pp. 361-368.

Harris, Z. 1985. Distributional Structure. In: Katz, J. J. (ed.), The Philosophy of Linguistics. New York: Oxford University Press. pp. 26-47.

Hindle, D. 1990. Noun classification from predicate-argument structures. In Proceedings of ACL-90. pp. 268–275. Pittsburgh, PA.

Hofmann, T. 1999. Probabilistic Latent Semantic Indexing. In *Proceedings of SIGIR-99*. pp. 50–57, Berkeley, California.

Kanerva, P. 1993. Sparse distributed memory and related models. pp. 50-76.

Lapata, M. and Keller, F., 2005. Web-based Models for Natural Language Processing, In ACM Transactions on Speech and Language Processing (TSLP), 2(1).

Lee, Lillian. 1999. Measures of Distributional Similarity. In Proceedings of ACL-93. pp. 25-32. College Park, MD.

Lin, D. 1998. Automatic retrieval and clustering of similar words. In Proceedings of COLING/ACL-98. pp. 768–774. Montreal, Canada.

Lund, K., and Burgess, C. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, 28(2):203–208.

McCallum, A. and Li, W. Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Enhanced Lexicons. In *Proceedings of CoNLL-03*.

McQueen, J. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematics, Statistics and Probability*, 1:281–298.

Paşca, M. 2007a. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of CIKM-07*. pp. 683-690.

Paşca, M. 2007b. Organizing and Searching the World Wide Web of Facts – Step Two: Harnessing the Wisdom of the Crowds. In *Proceedings of WWW-07*.

Paşca, M. and Durme, B.J. 2008. Weakly-supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs. *In Proceedings of ACL-08*.

Paşca, M.; Lin, D.; Bigham, J.; Lifchits, A.; Jain, A. 2006. Names and Similarities on the Web: Fast Extraction in the Fast Lane. In *Proceedings of ACL-2006*. pp. 113-120.

Pantel, P. and Lin, D. 2002. Discovering Word Senses from Text. In *Proceedings of KDD-02*. pp. 613-619. Edmonton, Canada.

Pantel, P.; Ravichandran, D; Hovy, E.H. 2004. Towards terascale knowledge acquisition. *In proceedings of COLING-04*. pp 771-777.

Ravichandran, D.; Pantel, P.; and Hovy, E. 2005. Randomized algorithms and NLP: Using locality sensitive hash function for high speed noun clustering. In *Proceedings of ACL-05*. pp. 622-629.

Riloff, E. and Jones, R. 1999 Learning Dictionaries for Information Extraction by Multi-Level Boostrapping. In *Proceedings of AAAI/IAAAI-99*.

Riloff, E. and Shepherd, J. 1997. A corpus-based approach for building semantic lexicons. In *Proceedings of EMNLP-97.*

Rychlý, P. and Kilgarriff, A. 2007. An efficient algorithm for building a distributional thesaurus (and other Sketch Engine developments). In *Proceedings of ACL-07*, demo sessions. Prague, Czech Republic.

Sarawagi, S. and Kirpal, A. 2004. Efficient set joins on similarity predicates. In *Proceedings of SIGMOD '04*. pp. 74 –754. New York, NY.

Sarmento, L.; Jijkuon, V.; de Rijke, M.; and Oliveira, E. 2007. "More like these": growing entity classes from seeds. In *Proceedings of CIKM-07*. pp. 959-962. Lisbon, Portugal.

Turney, P. D., and Littman, M. L. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems, 21(4)*.

Wang, R.C. and Cohen, W.W. 2008. Iterative Set Expansion of Named Entities using the Web. In *Proceedings of ICDM 2008*. Pisa, Italy.

Wang. R.C. and Cohen, W.W. 2007 Language-Independent Set Expansion of Named Entities Using the Web. In *Proceedings of ICDM-07*.

Yuret, D., and Yatbaz, M. A. 2009. The noisy channel model for unsupervised word sense disambiguation. *Computational Linguistics*. Under review.