Sparse Information Extraction: Unsupervised Language Models to the Rescue

Doug Downey, Stefan Schoenmackers, and Oren Etzioni
Turing Center, Department of Computer Science and Engineering
University of Washington, Box 352350
Seattle, WA 98195, USA
{ddowney,stef,etzioni}@cs.washington.edu

Abstract

Even in a massive corpus such as the Web, a substantial fraction of extractions appear infrequently. This paper shows how to assess the correctness of *sparse* extractions by utilizing unsupervised language models. The REALM system, which combines HMM-based and n-gram-based language models, ranks candidate extractions by the likelihood that they are correct. Our experiments show that REALM reduces extraction error by 39%, on average, when compared with previous work.

Because REALM pre-computes language models based on its corpus and does not require *any* hand-tagged seeds, it is far more scalable than approaches that learn models for each individual relation from hand-tagged data. Thus, REALM is ideally suited for open information extraction where the relations of interest are not specified in advance and their number is potentially vast.

1 Introduction

Information Extraction (IE) from text is far from infallible. In response, researchers have begun to exploit the redundancy in massive corpora such as the Web in order to assess the veracity of extractions (*e.g.*, (Downey et al., 2005; Etzioni et al., 2005; Feldman et al., 2006)). In essence, such methods utilize *extraction patterns* to generate candidate extractions (*e.g.*, "Istanbul") and then assess each candidate by computing co-occurrence statistics between the extraction and words or phrases indicative of class membership (*e.g.*, "cities such as").

However, Zipf's Law governs the distribution of extractions. Thus, even the Web has limited redundancy for less prominent instances of relations. Indeed, 50% of the extractions in the data sets employed by (Downey et al., 2005) appeared only once. As a result, Downey *et al.*'s model, and related methods, had no way of assessing which extraction is more likely to be correct for fully half of the extractions. This problem is particularly acute when moving beyond unary relations. We refer to this challenge as the task of *assessing sparse extractions*.

This paper introduces the idea that *language modeling* techniques such as n-gram statistics (Manning and Schütze, 1999) and HMMs (Rabiner, 1989) can be used to effectively assess sparse extractions. The paper introduces the REALM system, and highlights its unique properties. Notably, REALM does not require *any* hand-tagged seeds, which enables it to scale to *Open IE*—extraction where the relations of interest are not specified in advance, and their number is potentially vast (Banko et al., 2007).

REALM is based on two key hypotheses. The *KnowItAll hypothesis* is that extractions that occur more frequently in distinct sentences in the corpus are more likely to be correct. For example, the hypothesis suggests that the argument pair (`Giuliani`, `New York`) is relatively likely to be appropriate for the `Mayor` relation, simply because this pair is extracted for the `Mayor` relation relatively frequently. Second, we employ an instance of the *distributional hypothesis* (Harris, 1985), which

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 696–703,
Prague, Czech Republic, June 2007.

can be phrased as follows: different instances of the same semantic relation tend to appear in similar textual contexts. We assess sparse extractions by comparing the contexts in which they appear to those of more common extractions. Sparse extractions whose contexts are more similar to those of common extractions are judged more likely to be correct based on the conjunction of the KnowItAll and the distributional hypotheses.

The contributions of the paper are as follows:

- The paper introduces the insight that the subfield of language modeling provides unsupervised methods that can be leveraged to assess sparse extractions. These methods are more scalable than previous assessment techniques, and require no hand tagging whatsoever.
- The paper introduces an HMM-based technique for checking whether two arguments are of the proper type for a relation.
- The paper introduces a *relational* n-gram model for the purpose of determining whether a sentence that mentions multiple arguments actually expresses a particular relationship between them.
- The paper introduces a novel language-modeling system called REALM that combines both HMM-based models and relational n-gram models, and shows that REALM reduces error by an average of 39% over previous methods, when applied to sparse extraction data.

The remainder of the paper is organized as follows. Section 2 introduces the IE assessment task, and describes the REALM system in detail. Section 3 reports on our experimental results followed by a discussion of related work in Section 4. Finally, we conclude with a discussion of scalability and with directions for future work.

2 IE Assessment

This section formalizes the *IE assessment* task and describes the REALM system for solving it. An IE assessor takes as input a list of candidate extractions meant to denote instances of a relation, and outputs a *ranking* of the extractions with the goal that correct extractions rank higher than incorrect ones. A *correct* extraction is defined to be a true instance of the relation mentioned in the input text.

More formally, the list of candidate extractions for a relation R is denoted as $E_R = \{(a_1, b_1), \ldots, (a_m, b_m)\}$. An extraction (a_i, b_i) is an ordered pair of strings. The extraction is *correct* if and only if the relation R holds between the arguments named by a_i and b_i. For example, for $R =$ `Headquartered`, a pair (a_i, b_i) is correct *iff* there exists an organization a_i that is in fact headquartered in the location b_i.[1]

E_R is generated by applying an extraction mechanism, typically a set of extraction "patterns", to each sentence in a corpus, and recording the results. Thus, many elements of E_R are identical extractions derived from different sentences in the corpus.

This task definition is notable for the minimal inputs required—IE assessment does not require knowing the relation name nor does it require hand-tagged seed examples of the relation. Thus, an IE Assessor is applicable to Open IE.

2.1 System Overview

In this section, we describe the REALM system, which utilizes language modeling techniques to perform IE Assessment.

REALM takes as input a set of extractions E_R, and outputs a ranking of those extractions. The algorithm REALM follows is outlined in Figure 1. REALM begins by automatically selecting from E_R a set of *bootstrapped seeds* S_R intended to serve as correct examples of the relation R. REALM utilizes the KnowItAll hypothesis, setting S_R equal to the h elements in E_R extracted most frequently from the underlying corpus. This results in a noisy set of seeds, but the methods that use these seeds are noise tolerant.

REALM then proceeds to rank the remaining (non-seed) extractions by utilizing two language-modeling components. An *n-gram language model* is a probability distribution $P(w_1, ..., w_n)$ over consecutive word sequences of length n in a corpus. Formally, if we assume a seed (s_1, s_2) is a correct extraction of a relation R, the distributional hypothesis states that the *context distribution* around the seed extraction, $P(w_1, ..., w_n | w_i = s_1, w_j = s_2)$ for $1 \leq i, j \leq n$ tends to be "more similar" to

[1]For clarity, our discussion focuses on relations between pairs of arguments. However, the methods we propose can be extended to relations of any arity.

$P(w_1, ..., w_n | w_i = e_1, w_j = e_2)$ when the extraction (e_1, e_2) is correct. Naively comparing context distributions is problematic, however, because the arguments to a relation often appear separated by several intervening words. In our experiments, we found that when relation arguments appear together in a sentence, 75% of the time the arguments are separated by at least three words. This implies that n must be large, and for sparse argument pairs it is not possible to estimate such a large language model accurately, because the number of modeling parameters is proportional to the vocabulary size raised to the nth power. To mitigate sparsity, REALM utilizes smaller language models in its two components as a means of "backing-off" from estimating context distributions explicitly, as described below.

First, REALM utilizes an HMM to estimate whether each extraction has arguments of the proper type for the relation. Each relation R has a set of types for its arguments. For example, the relation `AuthorOf(a, b)` requires that its first argument be an author, and that its second be some kind of written work. Knowing whether extracted arguments are of the proper type for a relation can be quite informative for assessing extractions. The challenge is, however, that this type information is *not* given to the system since the relations (and the types of the arguments) are not known in advance. REALM solves this problem by comparing the distributions of the seed arguments and extraction arguments. Type checking mitigates data sparsity by leveraging *every* occurrence of the individual extraction arguments in the corpus, rather than only those cases in which argument pairs occur near each other.

Although argument type checking is invaluable for extraction assessment, it is not sufficient for extracting relationships between arguments. For example, an IE system using only type information might determine that `Intel` is a corporation and that `Seattle` is a city, and therefore erroneously conclude that `Headquartered(Intel, Seattle)` is correct. Thus, REALM's second step is to employ an n-gram-based language model to assess whether the extracted arguments share the appropriate relation. Again, this information is not given to the system, so REALM compares the context distributions of the extractions to those of the seeds. As described in Section 2.3, REALM employs a relational n-gram language model in order to accurately compare context distributions when extractions are sparse.

```
REALM(Extractions E_R = {e_1, ..., e_m})
  S_R = the h most frequent extractions in E_R
  U_R = E_R - S_R
  TypeRankings(U_R) ← HMM-T(S_R, U_R)
  RelationRankings(U_R) ← REL-GRAMS(S_R, U_R)
  return a ranking of E_R with the elements of S_R at the
    top (ranked by frequency) followed by the elements of
    U_R = {u_1, ..., u_{m-h}} ranked in ascending order of
    TypeRanking(u_i) * RelationRanking(u_i).
```

Figure 1: Pseudocode for REALM at run-time. The language models used by the HMM-T and REL-GRAMS components are constructed in a preprocessing step.

REALM executes the type checking and relation assessment components separately; each component takes the seed and non-seed extractions as arguments and returns a ranking of the non-seeds. REALM then combines the two components' assessments into a single ranking. Although several such combinations are possible, REALM simply ranks the extractions in ascending order of the product of the ranks assigned by the two components. The following subsections describe REALM's two components in detail.

We identify the proper nouns in our corpus using the LEX method (Downey et al., 2007). In addition to locating the proper nouns in the corpus, LEX also concatenates each multi-token proper noun (*e.g.*,`Los Angeles`) together into a single token. Both of REALM's components construct language models from this tokenized corpus.

2.2 Type Checking with HMM-T

In this section, we describe our type-checking component, which takes the form of a Hidden Markov Model and is referred to as HMM-T. HMM-T ranks the set U_R of non-seed extractions, with a goal of ranking those extractions with arguments of proper type for R above extractions containing type errors. Formally, let U_{Ri} denote the set of the ith arguments of the extractions in U_R. Let S_{Ri} be defined similarly for the seed set S_R.

Our type checking technique exploits the distributional hypothesis—in this case, the intuition that

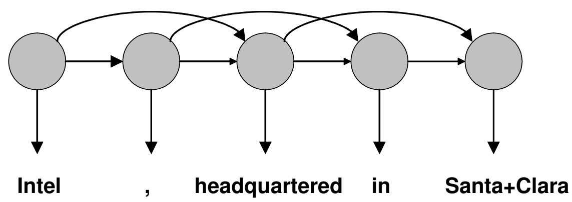

Figure 2: Graphical model employed by HMM-T. Shown is the case in which k = 2. Corpus pre-processing results in the proper noun `Santa Clara` being concatenated into a single token.

extraction arguments in U_{Ri} of the proper type will likely appear in contexts similar to those in which the seed arguments S_{Ri} appear. In order to identify terms that are distributionally similar, we train a probabilistic generative Hidden Markov Model (HMM), which treats each token in the corpus as generated by a single hidden state variable. Here, the hidden states take integral values from $\{1, \ldots, T\}$, and each hidden state variable is itself generated by some number k of previous hidden states.[2] Formally, the joint distribution of the corpus, represented as a vector of tokens $\mathbf{w}$, given a corresponding vector of states $\mathbf{t}$ is:

$$P(\mathbf{w}|\mathbf{t}) = \prod_i P(w_i|t_i)P(t_i|t_{i-1}, \ldots, t_{i-k}) \quad (1)$$

The distributions on the right side of Equation 1 can be learned from a corpus in an unsupervised manner, such that words which are distributed similarly in the corpus tend to be generated by similar hidden states (Rabiner, 1989). The generative model is depicted as a Bayesian network in Figure 2. The figure also illustrates the one way in which our implementation is distinct from a standard HMM, namely that proper nouns are detected *a priori* and modeled as single tokens (*e.g.*, `Santa Clara` is generated by a single hidden state). This allows the type checker to compare the state distributions of different proper nouns directly, even when the proper nouns contain differing numbers of words.

To generate a ranking of U_R using the learned HMM parameters, we rank the arguments e_i according to how similar their *state distributions* $P(t|e_i)$ are to those of the seed arguments.[3] Specifically, we define a function:

$$f(e) = \sum_{e_i \in e} KL\left(\frac{\sum_{w' \in S_{Ri}} P(t|w')}{|S_{Ri}|}, P(t|e_i)\right) \quad (2)$$

where KL represents KL divergence, and the outer sum is taken over the arguments e_i of the extraction e. We rank the elements of U_R in ascending order of $f(e)$.

HMM-T has two advantages over a more traditional type checking approach of simply counting the number of times in the corpus that each extraction appears in a context in which a seed also appears (*cf.* (Ravichandran et al., 2005)). The first advantage of HMM-T is efficiency, as the traditional approach involves a computationally expensive step of retrieving the potentially large set of contexts in which the extractions and seeds appear. In our experiments, using HMM-T instead of a context-based approach results in a 10-50x reduction in the amount of data that is retrieved to perform type checking. Secondly, on sparse data HMM-T has the potential to improve type checking accuracy. For example, consider comparing `Pickerington`, a sparse candidate argument of the type `City`, to the seed argument `Chicago`, for which the following two phrases appear in the corpus:

(i) "Pickerington, Ohio"
(ii) "Chicago, Illinois"

In these phrases, the textual contexts surrounding `Chicago` and `Pickerington` are not identical, so to the traditional approach these contexts offer no evidence that `Pickerington` and `Chicago` are of the same type. For a sparse token like `Pickerington`, this is problematic because the token may *never* occur in a context that precisely matches that of a seed. In contrast, in the HMM, the non-sparse tokens `Ohio` and `Illinois` are likely to have similar state distributions, as they are both the names of U.S. States. Thus, in the state space employed by the HMM, the contexts in phrases (i) and (ii) are in fact quite similar, allowing HMM-T to detect that `Pickerington` and `Chicago` are likely of the same type. Our experiments quantify the performance improvements that HMM-T of-

[2]Our implementation makes the simplifying assumption that each sentence in the corpus is generated independently.

[3]The distribution $P(t|e_i)$ for any e_i can be obtained from the HMM parameters using Bayes Rule.

fers over the traditional approach for type checking sparse data.

The time required to learn HMM-T's parameters scales proportional to T^{k+1} times the corpus size. Thus, for tractability, HMM-T uses a relatively small state space of $T = 20$ states and a limited k value of 3. While these settings are sufficient for type checking (*e.g.*, determining that `Santa Clara` is a city) they are too coarse-grained to assess relations between arguments (*e.g.*, determining that `Santa Clara` is the particular city in which `Intel` is headquartered). We now turn to the REL-GRAMS component, which performs the latter task.

2.3 Relation Assessment with REL-GRAMS

REALM's relation assessment component, called REL-GRAMS, tests whether the extracted arguments have a desired relationship, but given REALM's minimal input it has no *a priori* information about the relationship. REL-GRAMS relies instead on the distributional hypothesis to test each extraction.

As argued in Section 2.1, it is intractable to build an accurate language model for context distributions surrounding sparse argument pairs. To overcome this problem, we introduce *relational* n-gram models. Rather than simply modeling the context distribution around a given argument, a relational n-gram model specifies separate context distributions for an arguments *conditioned on* each of the other arguments with which it appears. The relational n-gram model allows us to estimate context distributions for pairs of arguments, even when the arguments do not appear together within a fixed window of n words. Further, by considering only consecutive argument pairs, the number of distinct argument pairs in the model grows at most linearly with the number of sentences in the corpus. Thus, the relational n-gram model can scale.

Formally, for a pair of arguments (e_1, e_2), a relational n-gram model estimates the distributions $P(w_1, ..., w_n | w_i = e_1, e_1 \leftrightarrow e_2)$ for each $1 \leq i \leq n$, where the notation $e_1 \leftrightarrow e_2$ indicates the event that e_2 is the next argument to either the right or the left of e_1 in the corpus.

REL-GRAMS begins by building a relational n-gram model of the arguments in the corpus. For notational convenience, we represent the model's distributions in terms of "context vectors" for each pair of arguments. Formally, for a given sentence containing arguments e_1 and e_2 consecutively, we define a *context* of the ordered pair (e_1, e_2) to be any window of n tokens around e_1. Let $C = \{c_1, c_2, ..., c_{|C|}\}$ be the set of all contexts of all argument pairs found in the corpus.[4] For a pair of arguments (e_j, e_k), we model their relationship using a $|C|$ dimensional context vector $v_{(e_j, e_k)}$, whose i-th dimension corresponds to the number of times context c_i occurred with the pair (e_j, e_k) in the corpus. These context vectors are similar to document vectors from Information Retrieval (IR), and we leverage IR research to compare them, as described below.

To assess each extraction, we determine how similar its context vector is to a canonical seed vector (created by summing the context vectors of the seeds). While there are many potential methods for determining similarity, in this work we rank extractions by decreasing values of the BM25 distance metric. BM25 is a TF-IDF variant introduced in TREC-3(Robertson et al., 1992), which outperformed both the standard cosine distance and a smoothed KL divergence on our data.

3 Experimental Results

This section describes our experiments on IE assessment for sparse data. We start by describing our experimental methodology, and then present our results. The first experiment tests the hypothesis that HMM-T outperforms an n-gram-based method on the task of type checking. The second experiment tests the hypothesis that REALM outperforms multiple approaches from previous work, and also outperforms each of its HMM-T and REL-GRAMS components taken in isolation.

3.1 Experimental Methodology

The corpus used for our experiments consisted of a sample of sentences taken from Web pages. From an initial crawl of nine million Web pages, we selected sentences containing relations between proper nouns. The resulting text corpus consisted of about

[4] Pre-computing the set C requires identifying in advance the potential relation arguments in the corpus. We consider the proper nouns identified by the LEX method (see Section 2.1) to be the potential arguments.

three million sentences, and was tokenized as described in Section 2. For tractability, before and after performing tokenization, we replaced each token occurring fewer than five times in the corpus with one of two "unknown word" markers (one for capitalized words, and one for uncapitalized words). This preprocessing resulted in a corpus containing about sixty-five million total tokens, and 214,787 unique tokens.

We evaluated performance on four relations: `Conquered`, `Founded`, `Headquartered`, and `Merged`. These four relations were chosen because they typically take proper nouns as arguments, and included a large number of sparse extractions. For each relation R, the candidate extraction list E_R was obtained using TEXTRUNNER (Banko et al., 2007). TEXTRUNNER is an IE system that computes an index of all extracted relationships it recognizes, in the form of (object, predicate, object) triples. For each of our target relations, we executed a single query to the TEXTRUNNER index for extractions whose predicate contained a phrase indicative of the relation (*e.g.*, "founded by", "headquartered in"), and the results formed our extraction list. For each relation, the 10 most frequent extractions served as bootstrapped seeds. All of the non-seed extractions were sparse (no argument pairs were extracted more than twice for a given relation). These test sets contained a total of 361 extractions.

3.2 Type Checking Experiments

As discussed in Section 2.2, on sparse data HMM-T has the potential to outperform type checking methods that rely on textual similarities of context vectors. To evaluate this claim, we tested the HMM-T system against an N-GRAMS type checking method on the task of type-checking the arguments to a relation. The N-GRAMS method compares the context vectors of extractions in the same way as the REL-GRAMS method described in Section 2.3, but is not relational (N-GRAMS considers the distribution of each extraction argument independently, similar to HMM-T). We tagged an extraction as type correct *iff* both arguments were valid for the relation, ignoring whether the relation held between the arguments.

The results of our type checking experiments are shown in Table 1. For all types, HMM-T outperforms N-GRAMS, and HMM-T reduces error (measured in missing area under the precision/recall curve) by 46%. The performance difference on each relation is statistically significant ($p < 0.01$, two-sampled t-test), using the methodology for measuring the standard deviation of area under the precision/recall curve given in (Richardson and Domingos, 2006). N-GRAMS, like REL-GRAMS, employs the BM-25 metric to measure distributional similarity between extractions and seeds. Replacing BM-25 with cosine distance cuts HMM-T's advantage over N-GRAMS, but HMM-T's error rate is still 23% lower on average.

Type	HMM-T	N-GRAMS
`Conquered`	**0.917**	0.767
`Founded`	**0.827**	0.636
`Headquartered`	**0.734**	0.589
`Merged`	**0.920**	0.854
Average	**0.849**	0.712

Table 1: Type Checking Performance. Listed is area under the precision/recall curve. HMM-T outperforms N-GRAMS for all relations, and reduces the error in terms of missing area under the curve by 46% on average.

3.3 Experiments with REALM

The REALM system combines the type checking and relation assessment components to assess extractions. Here, we test the ability of REALM to improve the ranking of a state of the art IE system, TEXTRUNNER. For these experiments, we evaluate REALM against the TEXTRUNNER frequency-based ordering, a pattern-learning approach, and the HMM-T and REL-GRAMS components taken in isolation. The TEXTRUNNER frequency-based ordering ranks extractions in decreasing order of their extraction frequency, and importantly, for our task this ordering is essentially equivalent to that produced by the "Urns" (Downey et al., 2005) and Pointwise Mutual Information (Etzioni et al., 2005) approaches employed in previous work.

The pattern-learning approach, denoted as PL, is modeled after Snowball (Agichtein, 2006). The algorithm and parameter settings for PL were those manually tuned for the `Headquartered` relation in previous work (Agichtein, 2005). A sensitivity analysis of these parameters indicated that the re-

	Conquered	Founded	Headquartered	Merged	Average
Avg. Prec.	0.698	0.578	0.400	0.742	0.605
TEXTRUNNER	0.738	0.699	0.710	0.784	0.733
PL	0.885	0.633	0.651	0.852	0.785
PL+ HMM-T	0.883	0.722	0.727	0.900	0.808
HMM-T	0.830	0.776	0.678	0.864	0.787
REL-GRAMS	**0.929 (39%)**	0.713	0.758	0.886	0.822
REALM	0.907 (19%)	**0.781 (27%)**	**0.810 (35%)**	**0.908 (38%)**	**0.851 (39%)**

Table 2: Performance of REALM for assessment of sparse extractions. Listed is area under the precision/recall curve for each method. In parentheses is the percentage reduction in error over the strongest baseline method (TEXTRUNNER or PL) for each relation. "Avg. Prec." denotes the fraction of correct examples in the test set for each relation. REALM outperforms its REL-GRAMS and HMM-T components taken in isolation, as well as the TEXTRUNNER and PL systems from previous work.

sults are sensitive to the parameter settings. However, we found no parameter settings that performed significantly better, and many settings performed significantly worse. As such, we believe our results reasonably reflect the performance of a pattern learning system on this task. Because PL performs relation assessment, we also attempted combining PL with HMM-T in a hybrid method (PL+ HMM-T) analogous to REALM.

The results of these experiments are shown in Table 2. REALM outperforms the TEXTRUNNER and PL baselines for all relations, and reduces the missing area under the curve by an average of 39% relative to the strongest baseline. The performance differences between REALM and TEXTRUNNER are statistically significant for all relations, as are differences between REALM and PL for all relations except Conquered ($p < 0.01$, two-sampled t-test). The hybrid REALM system also outperforms each of its components in isolation.

4 Related Work

To our knowledge, REALM is the first system to use language modeling techniques for IE Assessment.

Redundancy-based approaches to pattern-based IE assessment (Downey et al., 2005; Etzioni et al., 2005) require that extractions appear relatively frequently with a limited set of patterns. In contrast, REALM utilizes *all* contexts to build a model of extractions, rather than a limited set of patterns. Our experiments demonstrate that REALM outperforms these approaches on sparse data.

Type checking using *named-entity taggers* has been previously shown to improve the precision of pattern-based IE systems (Agichtein, 2005; Feldman et al., 2006), but the HMM-T type-checking component we develop differs from this work in important ways. Named-entity taggers are limited in that they typically recognize only small set of types (*e.g.*, ORGANIZATION, LOCATION, PERSON), and they require hand-tagged training data for each type. HMM-T, by contrast, performs type checking for *any* type. Finally, HMM-T does not require hand-tagged training data.

Pattern learning is a common technique for extracting and assessing sparse data (*e.g.* (Agichtein, 2005; Riloff and Jones, 1999; Paşca et al., 2006)). Our experiments demonstrate that REALM outperforms a pattern learning system closely modeled after (Agichtein, 2005). REALM is inspired by pattern learning techniques (in particular, both use the distributional hypothesis to assess sparse data) but is distinct in important ways. Pattern learning techniques require substantial processing of the corpus after the relations they assess have been specified. Because of this, pattern learning systems are unsuited to Open IE. Unlike these techniques, REALM pre-computes language models which allow it to assess extractions for arbitrary relations at run-time. In essence, pattern-learning methods run in time linear in the number of relations whereas REALM's run time is constant in the number of relations. Thus, REALM scales readily to large numbers of relations whereas pattern-learning methods do not.

A second distinction of REALM is that its type checker, unlike the named entity taggers employed in pattern learning systems (*e.g.*, Snowball), can be used to identify arbitrary types. A final distinction is that the language models REALM employs require fewer parameters and heuristics than pattern learning techniques.

Similar distinctions exist between REALM and a recent system designed to assess sparse extractions by bootstrapping a *classifier* for each target relation (Feldman et al., 2006). As in pattern learning, constructing the classifiers requires substantial processing after the target relations have been specified, and a set of hand-tagged examples per relation, making it unsuitable for Open IE.

5 Conclusions

This paper demonstrated that unsupervised language models, as embodied in the REALM system, are an effective means of assessing sparse extractions.

Another attractive feature of REALM is its scalability. Scalability is a particularly important concern for *Open Information Extraction*, the task of extracting large numbers of relations that are not specified in advance. Because HMM-T and REL-GRAMS both pre-compute language models, REALM can be queried efficiently to perform IE Assessment. Further, the language models are constructed independently of the target relations, allowing REALM to perform IE Assessment even when relations are not specified in advance.

In future work, we plan to develop a probabilistic model of the information computed by REALM. We also plan to evaluate the use of non-local context for IE Assessment by integrating document-level modeling techniques (*e.g.*, Latent Dirichlet Allocation).

Acknowledgements

This research was supported in part by NSF grants IIS-0535284 and IIS-0312988, DARPA contract NBCHD030010, ONR grant N00014-05-1-0185 as well as a gift from Google. The first author is supported by an MSR graduate fellowship sponsored by Microsoft Live Labs. We thank Michele Banko, Jeff Bilmes, Katrin Kirchhoff, and Alex Yates for helpful comments.

References

E. Agichtein. 2005. *Extracting Relations From Large Text Collections*. Ph.D. thesis, Department of Computer Science, Columbia University.

E. Agichtein. 2006. Confidence estimation methods for partially supervised relation extraction. In *SDM 2006*.

M. Banko, M. Cararella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the web. In *Procs. of IJCAI 2007*.

D. Downey, O. Etzioni, and S. Soderland. 2005. A Probabilistic Model of Redundancy in Information Extraction. In *Procs. of IJCAI 2005*.

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

O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

R. Feldman, B. Rosenfeld, S. Soderland, and O. Etzioni. 2006. Self-supervised relation extraction from the web. In *ISMIS*, pages 755–764.

Z. Harris. 1985. Distributional structure. In J. J. Katz, editor, *The Philosophy of Linguistics*, pages 26–47. New York: Oxford University Press.

C. D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*.

M. Paşca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. 2006. Names and similarities on the web: Fact extraction in the fast lane. In *Procs. of ACL/COLING 2006*.

L. R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

D. Ravichandran, P. Pantel, and E. H. Hovy. 2005. Randomized Algorithms and NLP: Using Locality Sensitive Hash Functions for High Speed Noun Clustering. In *Procs. of ACL 2005*.

M. Richardson and P. Domingos. 2006. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136.

E. Riloff and R. Jones. 1999. Learning Dictionaries for Information Extraction by Multi-level Boot-strapping. In *Procs. of AAAI-99*, pages 1044–1049.

S. E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. 1992. Okapi at TREC-3. In *Text REtrieval Conference*, pages 21–30.