Large Scale Relation Detection[*]

Chris Welty and **James Fan** and **David Gondek** and **Andrew Schlaikjer**

IBM Watson Research Center · 19 Skyline Drive · Hawthorne, NY 10532, USA
{welty, fanj, dgondek, ahschlai}@us.ibm.com

Abstract

We present a technique for reading sentences and producing sets of hypothetical relations that the sentence may be expressing. The technique uses large amounts of instance-level background knowledge about the relations in order to gather statistics on the various ways the relation may be expressed in language, and was inspired by the observation that half of the linguistic forms used to express relations occur very infrequently and are simply not considered by systems that use too few seed examples. Some very early experiments are presented that show promising results.

1 Introduction

We are building a system that learns to read in a new domain by applying a novel combination of natural language processing, machine learning, knowledge representation and reasoning, information retrieval, data mining, etc. techniques in an integrated way. Central to our approach is the view that all parts of the system should be able to interact during any level of processing, rather than a pipeline view in which certain parts of the system only take as input the results of other parts, and thus cannot influence those results. In this paper we discuss a particular case of that idea, using large knowledge bases hand in hand with natural language processing to improve the quality of relation detection. Ultimately we define reading as representing natural language text in a way that integrates background knowledge and inference, and thus are doing the relation detection to better integrate text with pre-existing knowledge, however that should not (and does not) prevent us from using what knowledge we have to influence that integration along the way.

[*] Research supported in part by DARPA MRP Grant FA8750-09-C0172

2 Background

The most obvious points of interaction between NLP and KR systems are named entity tagging and other forms of type instance extraction. The second major point of interaction is relation extraction, and while there are many kinds of relations that may be detected (e.g. syntactic relations such as modifiers and verb subject/object, equivalence relations like coreference or nicknames, event frame relations such as participants, etc.), the kind of relations that reading systems need to extract to support domain-specific reasoning tasks are relations that are known to be expressed in supporting knowledge-bases. We call these relations semantic relations in this paper. Compared to entity and type detection, extraction of semantic relations is significantly harder. In our work on bridging the NLP-KR gap, we have observed several aspects of what makes this task difficult, which we discuss below.

2.1 Keep reading

Humans do not read and understand text by first recognizing named entities, giving them types, and then finding a small fixed set of relations between them. Rather, humans start with the first sentence and build up a representation of what they read that expands and is refined during reading. Furthermore, humans do not "populate databases" by reading; knowledge is not only a product of reading, it is an *integral part* of it. We require knowledge during reading in order to understand what we read.

One of the central tenets of our machine reading system is the notion that reading is not performed on sentences in isolation. Often, problems in NLP can be resolved by simply waiting for the next sentence, or remembering the results from the previous, and incorporating background or domain specific knowledge. This includes parse ambiguity, coreference, typing of named entities, etc. We call this the *Keep Reading* principle.

Keep reading applies to relation extraction as well. Most relation extraction systems are implemented such that a single interpretation is forced on a sentence, based only on features of the sentence itself. In fact, this has been a shortcoming of many NLP systems in the past. However, when you apply the Keep Reading principle, multiple hypotheses from different parts of the NLP pipeline are maintained, and decisions are deferred until there is enough evidence to make a high confidence choice between competing hypotheses. Knowledge, such as those entities already known to participate in a relation and how that relation was expressed, can and should be part of that evidence. We will present many examples of the principle in subsequent sections.

2.2 Expressing relations in language

Due to the flexibility and expressive power of natural language, a specific type of semantic relation can usually be expressed in language in a myriad of ways. In addition, semantic relations are often implied by the expression of other relations. For example, all of the following sentences more or less express the same relation between an actor and a movie: (1) "Elijah wood starred in Lord of the Rings: The Fellowship of the Ring", (2) "Lord of the Rings: The Fellowship of the Ring's Elijah Wood, ...", and(3) "Elijah Wood's coming of age was clearly his portrayal of the dedicated and noble hobbit that led the eponymous fellowship from the first episode of the Lord of the Rings trilogy." No human reader would have any trouble recognizing the relation, but clearly this variability of expression presents a major problem for machine reading systems.

To get an empirical sense of the variability of natural language used to express a relation, we studied a few semantic relations and found sentences that expressed that relation, extracted simple *patterns* to account for how the relation is expressed between two arguments, mainly by removing the relation arguments (e.g. "Elijah Wood" and "Lord of the Rings: The Fellowship of the Ring" above) and replacing them with variables. We then counted the number of times each pattern was used to express the relation, producing a recognizable *very long tail* shown in Figure 1 for the top 50 patterns expressing the acted-in-movie relation in 17k sentences. More sophisticated pattern generalization (as discussed in later sections) would significantly fatten the head, bringing it closer to the traditional 50% of the area under the curve, but no amount of generalization will eliminate the tail. The patterns become increasingly esoteric, such as "The movie Death Becomes Her features a brief sequence in which Bruce Willis and Goldie Hawn's characters plan Meryl Streep's character's death by sending her car off of a cliff on Mulholland Drive," or "The best known Hawksian woman is probably Lauren Bacall, who iconically played the type opposite Humphrey Bogart in To Have and Have Not and The Big Sleep."

2.3 What relations matter

We do not consider relation extraction to be an end in and of itself, but rather as a component in larger systems that perform some task requiring interoperation between language- and knowledge-based components. Such larger tasks can include question answering, medical diagnosis, intelligence analysis, museum curation, etc. These tasks have evaluation criteria that go beyond measuring relation extraction results. The first step in applying relation detection to these larger tasks is analysis to determine what relations matter for the task and domain.

There are a number of manual and semi-automatic ways to perform such analysis. Repeating the theme of this paper, which is to use pre-existing knowledge-bases as resources, we performed this analysis using freebase and a set of 20k question-answer pairs representing our task domain. For each question, we formed tuples of each entity name in the question (QNE) with the answer, and found all

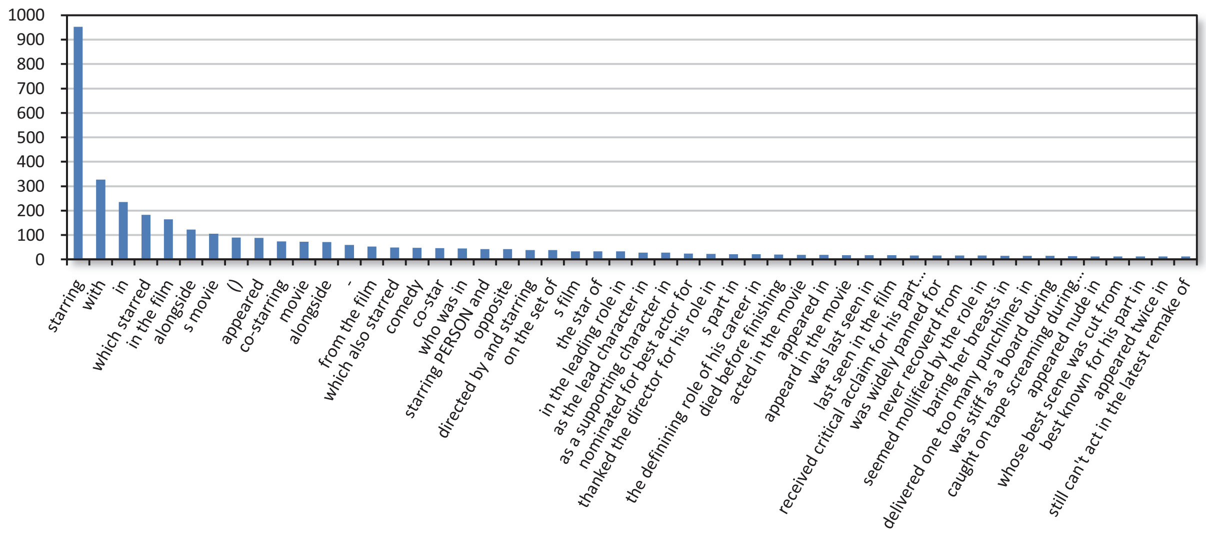

Figure 1: Pattern frequency for *acted-in-movie* relation for 17k sentences.

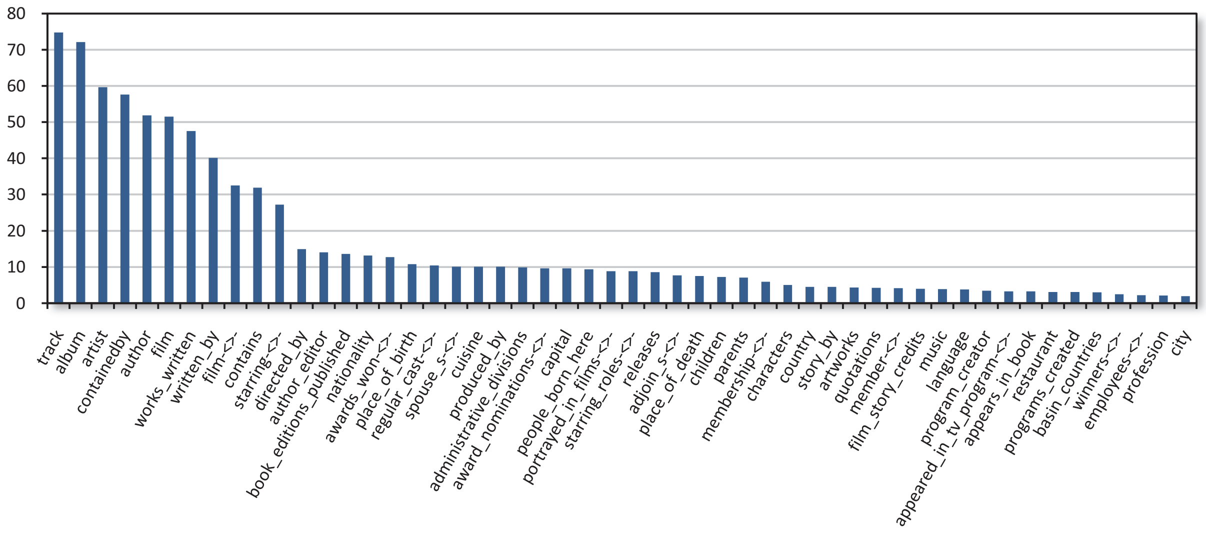

Figure 2: Relative frequency for top 50 relations in 20K question-answer pairs.

the relations in the KB connecting the entities. We kept a count for each relation of how often it connected a QNE to an answer. Of course we don't actually know for sure that the relation is the one being asked, but the intuition is that if the amount of data is big enough, you will have at least a ranked list of which relations are the most frequent.

Figure 2 shows the ranking for the top 50 relations. Note that, even when restricted to the top 50 relations, the graph has no head, it is basically all tail; The top 50 relations cover about 15% of the domain. In smaller, manual attempts to determine the most frequent relations in our domain, we had a similar result. What this means is that supporting even the top 50 relations with perfect recall covers about 15% of the questions. It is possible, of course, to narrow the domain and restrict the relations that can be queried–this is what database systems do. For reading, however, the results are the same. A reading system requires the ability to recognize hundreds of relations to have any significant impact on understanding.

2.4 Multi-relation learning on many seeds

The results shown in Figure 1 and Figure 2 confirmed much of the analysis and experiences we'd had in the past trying to apply relation extraction in the traditional way to natural language problems like question answering, building concept graphs from intelligence reports, semantic search, etc. Either by training machine learning algorithms on manually annotated data or by manually crafting finite-state transducers, relation detection is faced by this two-fold problem: the per-relation extraction hits a wall around 50% recall, and each relation itself occurs infrequently in the data.

This apparent futility of relation extraction led us to rethink our approach. First of all, the very long tail for relation patterns led us to consider how to pick up the tail. We concluded that to do so would require many more examples of the relation, but where can we get them? In the world of linked-data, huge instance-centered knowledge-bases are rapidly growing and spreading on the semantic web[1]. Resources like DBPedia, Freebase, IMDB, Geonames, the Gene Ontology, etc., are making available RDF-based data about a number of domains. These sources of structured knowledge can provide a large number of seed tuples for many different relations. This is discussed further below.

Furthermore, the all-tail nature of relation coverage led us to consider performing relation extraction on multiple relations at once. Some promising results on multi-relation learning have already been reported in (Carlson et al., 2009), and the data sources mentioned above give us many more than just the handful of seed instances used in those experiments. The idea of learning multiple relations at once also fits with our keep reading principle - multiple relation hypotheses may be annotated between the same arguments, with further evidence helping to disambiguate them.

3 Approach

One common approach to relation extraction is to start with *seed tuples* and find sentences that contain mentions of both elements of the tuple. From each such sentence a pattern is generated using at minimum universal generalization (replace the tuple elements with variables), though adding any form of generalization here can significantly improve recall. Finally, evaluate the patterns by applying them to text and evaluating the precision and recall of the tuples extracted by the patterns. Our approach, called *Large Scale Relation Detection* (LSRD), differs in three important ways:

1. We start with a knowledge-base containing a large number (thousands to millions) of tuples encoding relation instances of various types. Our hypothesis is that *only a large number of examples can possibly account for the long tail.*
2. We do not learn one relation at a time, but rather, associate a pattern with a set of relations whose tuples appear in that pattern. Thus, when a pattern is matched to a sentence during reading, each relation in its set of associated relations is posited as a hypothetical interpretation of the sentence, to be supported or refuted by further reading.
3. We use the knowledge-base as an oracle to determine negative examples of a relation. As a result the technique is semi-supervised; it requires no human intervention but does require reliable knowledge-bases as input–these knowledge-bases are readily available today.

Many relation extraction techniques depend on a prior step of named entity recognition (NER) and typing, in order to identify potential arguments. However, this limits recall to the recall of the NER step. In our approach patterns can match on any noun phrase, and typing of these NPs is simply another form of evidence.

All this means our approach is not relation extraction per se, it typically does not make conclusions about a relation in a sentence, but extracts hypotheses to be resolved by other parts of our reading system.

In the following sections, we elaborate on the technique and some details of the current implementation.

3.1 Basic pipeline

The two principle inputs are a corpus and a knowledge-base (KB). For the experiments below, we used the English Gigaword corpus[2] extended with Wikipedia and other news sources, and IMDB, DBPedia, and Freebase KBs, as shown. The intent is

[1] `http://linkeddata.org/`

[2] `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T05`

to run against a web-scale corpus and larger linked-data sets.

Input documents are sentence delimited, tokenized and parsed. The technique can benefit dramatically from coreference resolution, however in the experiments shown, this was not present. For each pair of *proper names* in a sentence, the names are looked up in the KB, and if they are related, a *pattern* is extracted from the sentence. At minimum, pattern extraction should replace the names with variables. Depending on how patterns are extracted, one pattern may be extracted per sentence, or one pattern may be extracted per pair of proper names in the sentence. Each pattern is associated with *all the relations* known in the KB between the two proper names. If the pattern has been extracted before, the two are merged by incrementing the associated relation counts. This phase, called *pattern induction*, is repeated for the entire corpus, resulting in a large set of patterns, each pattern associated with relations. For each ¡pattern, relation¿ pair, there is a count of the number of times that pattern appeared in the corpus with names that are in the relation according to the KB.

The pattern induction phase results in *positive counts*, i.e. the number of times a pattern appeared in the corpus with named entities known to be related in the KB. However, the induction phase does not exhaustively count the number of times each pattern appears in the corpus, as a pattern may appear with entities that are not known in the KB, or are not known to be related. The second phase, called *pattern training*, goes through the entire corpus again, trying to match induced patterns to sentences, binding any noun phrase to the pattern variables. Some attempt is made to resolve the noun phrase to something (most obviously, a name) that can be looked up in the KB, and for each relation associated with the pattern, if the two names are *not* in the relation according to the KB, the *negative count* for that relation in the matched pattern is incremented. The result of the pattern training phase is an updated set of ¡pattern, relation¿ pairs with *negative counts*.

The following example illustrates the basic processing. During induction, this sentence is encountered:

> Tom Cruise and co-star Nicole Kidman appeared together at the premier.

The proper names "Tom Cruise" and "Nicole Kidman" are recognized and looked up in the KB. We find instances in the KB with those names, and the following relations: `coStar(Tom Cruise, Nicole Kidman)`; `marriedTo(Tom Cruise, Nicole Kidman)`. We extract a pattern p_1: `?`x` and co-star ?`y` appeared together at the premier` in which all the names have been replace by variables, and the associations <p_1, `costar, 1, 0`> and <p_1, `marriedTo, 1, 0`> with positive counts and zero negative counts. Over the entire corpus, we'd expect the pattern to appear a few times and end up with final positive counts like <p_1, `coStar, 14, 0`> and <p_1, `marriedTo, 2, 0`>, indicating the pattern p_1 appeared 14 times in the corpus between names known to participate in the *coStar* relation, and twice between names known to participate in the *marriedTo* relation. During training, the following sentence is encountered that matches p_1:

> Tom Hanks and co-star Daryl Hannah appeared together at the premier.

The names "Tom Hanks" and "Daryl Hannah" are looked up in the KB and in this case only the relation *coStar* is found between them, so the *marriedTo* association is updated with a negative count: <p_1, `marriedTo, 2, -1`>. Over the entire corpus, we'd expect the counts to be something like <p_1, `costar, 14, -6`> and <p_1, `marriedTo, 2, -18`>.

This is a very simple example and it is difficult to see the value of the pattern training phase, as it may appear the negative counts could be collected during the induction phase. There are several reasons why this is not so. First of all, since the first phase only induces patterns between proper names that appear and are related within the KB, a sentence in the corpus matching the pattern would be missed if it did not meet that criteria but was encountered before the pattern was induced. Secondly, for reasons that are beyond the scope of this paper, having to do with our *Keep Reading* principle, the second phase does slightly more general matching: note that it matches noun phrases instead of proper nouns.

3.2 Candidate-instance matching

An obvious part of the process in both phases is taking strings from text and matching them against names or labels in the KB. We refer to the strings in the sentences as *candidate arguments* or simply *candidates*, and refer to *instances* in the KB as entities with associated attributes. For simplicity of discussion we will assume all KBs are in RDF, and thus all KB instances are nodes in a graph with unique identifiers (URIs) and arcs connecting them to other instances or primitive values (strings, numbers, etc.). A set of specially designated arcs, called *labels*, connect instances to strings that are understood to name the instances. The reverse lookup of entity identifiers via names referred to in the previous section requires searching for the labels that match a string found in a sentence and returning the instance identifier.

This step is so obvious it belies the difficultly of the matching process and is often overlooked, however in our experiments we have found candidate-instance matching to be a significant source of error. Problems include having many instances with the same or lexically similar names, slight variations in spelling especially with non-English names, inflexibility or inefficiency in string matching in KB implementations, etc. In some of our sources, names are also encoded as URLs. In the case of movie and book titles-two of the domains we experimented with-the titles seem almost as if they were designed specifically to befuddle attempts to automatically recognize them. Just about every English word is a book or movie title, including “It”, “Them”, “And”, etc., many years are titles, and just about every number under 1000. Longer titles are difficult as well, since simple lexical variations can prevent matching from succeeding, e.g. the Shakespeare play, *A Midsummer Night’s Dream* appears often as *Midsummer Night’s Dream*, *A Midsummer Night Dream*, and occasionally, in context, just *Dream*. When titles are not distinguished or delimited somehow, they can confuse parsing which may fail to recognize them as noun phrases. We eventually had to build dictionaries of multi-word titles to help parsing, but of course that was imperfect as well.

The problems go beyond the analogous ones in coreference resolution as the sources and technology themselves are different. The problems are severe enough that the candidate-instance matching problem contributes the most, of all components in this pipeline, to precision and recall failures. We have observed recall drops of as much as 15% and precision drops of 10% due to candidate-instance matching.

This problem has been studied somewhat in the literature, especially in the area of database record matching and coreference resolution (Michelson and Knoblock, 2007), but the experiments presented below use rudimentary solutions and would benefit significantly from improvements; it is important to acknowledge that the problem exists and is not as trivial as it appears at first glance.

3.3 Pattern representation

The basic approach accommodates any pattern representation, and in fact we can accommodate non pattern-based learning approaches, such as CRFs, as the primary hypothesis is principally concerned with the number of seed examples (scaling up initial set of examples is important). Thus far we have only experimented with two pattern representations: simple lexical patterns in which the known arguments are replaced in the sentence by variables (as shown in the example above), and patterns based on the spanning tree between the two arguments in a dependency parse, again with the known arguments replaced by variables. In our initial design we downplayed the importance of the pattern representation and especially generalization, with the belief that very large scale would remove the need to generalize. However, our initial experiments suggest that good pattern generalization would have a significant impact on recall, without negative impact on precision, which agrees with findings in the literature (Pantel and Pennacchiotti, 2006). Thus, these early results only employ rudimentary pattern generalization techniques, though this is an area we intend to improve. We discuss some more details of the lack of generalization below.

4 Experiment

In this section we present a set of very early proof of concept experiments performed using drastic simplifications of the LSRD design. We began, in fact, by

Relation	*Prec*	*Rec*	*F1*	*Tuples*	*Seeds*
imdb:actedIn	46.3	45.8	0.46	9M	30K
frb:authorOf	23.4	27.5	0.25	2M	2M
imdb:directorOf	22.8	22.4	0.22	700K	700K
frb:parentOf	68.2	8.6	0.16	10K	10K

Table 1: Precision and recall vs. number of tuples used for 4 freebase relations.

using single-relation experiments, despite the centrality of multiple hypotheses to our reading system, in order to facilitate evaluation and understanding of the technique. Our main focus was to gather data to support (or refute) the hypothesis that more relation examples would matter during pattern induction, and that using the KB as an oracle for training would work. Clearly, no KB is complete to begin with, and candidate-instance matching errors drop apparent coverage further, so we intended to explore the degree to which the KB's coverage of the relation impacted performance. To accomplish this, we examined four relations with different coverage characteristics in the KB.

4.1 Setup and results

The first relation we tried was the *acted-in-show* relation from IMDB; for convenience we refer to it as *imdb:actedIn*. An IMDB *show* is a movie, TV episode, or series. This relation has over 9M `<actor, show>` tuples, and its coverage was complete as far as we were able to determine. However, the version we used did not have a lot of name variations for actors. The second relation was the *author-of* relation from Freebase (*frb:authorOf*), with roughly 2M `<author, written-work>` tuples. The third relation was the *director-of-movie* relation from IMDB (*imdb:directorOf*), with 700k `<director,movie>` tuples. The fourth relation was the *parent-of* relation from Freebase (*frb:parentOf*), with roughly 10K `<parent, child>` tuples (mostly biblical and entertainment). Results are shown in Table 1.

The *imdb:actedIn* experiment was performed on the first version of the system that ran on 1 CPU and, due to resource constraints, was not able to use more than 30K seed tuples for the rule induction phase. However, the full KB (9M relation instances) was available for the training phase. With some manual effort, we selected tuples (actor-movie pairs) of popular actors and movies that we expected to appear most frequently in the corpus. In the other experiments, the full tuple set was available for both phases, but 2M tuples was the limit for the size of the KB in the implementation. With these promising preliminary results, we expect a full implementation to accommodate up to 1B tuples or more.

The evaluation was performed in decreasing degrees of rigor. The *imdb:actedIn* experiment was run against 20K sentences with roughly 1000 actor in movie relations and checked by hand. For the other three, the same sentences were used, but the ground truth was generated in a semi-automatic way by reusing the LSRD assumption that a sentence containing tuples in the relation expresses the relation, and then spot-checked manually. Thus the evaluation for these three experiments favors the LSRD approach, though spot checking revealed it is the precision and not the recall that benefits most from this, and all the recall problems in the ground truth (i.e. sentences that did express the relation but were not in the ground truth) were due to candidate-instance matching problems. An additional idiosyncrasy in the evaluation is that the sentences in the ground truth were actually questions, in which one of the arguments to the relation was the answer. Since the patterns were induced and trained on statements, there is a mismatch in style which also significantly impacts recall. Thus the precision and recall numbers should not be taken as general performance, but are useful only relative to each other.

4.2 Discussion

The results are promising, and we are continuing the work with a scalable implementation. Overall, the results seem to show a clear correlation between the number of seed tuples and relation extraction recall. However, the results do not as clearly support the many examples hypothesis as it may seem. When an actor and a film that actor starred in are mentioned in a sentence, it is very often the case that the sentence expresses that relation. However, this was less likely in the case of the parent-of relation, and as we considered other relations, we found a wide degree of variation. The borders relation between two countries, for example, is on the other extreme from actor-in-movie. Bordering nations often wage war, trade, suspend relations, deport refugees, support, oppose, etc. each other, so finding the two nations in a sentence together is not highly indicative of one relation or another. The director-of-movie relation was closer to acted-in-movie in this regard, and author-of a bit below that. The obvious next step to gather more data on the many examples hypothesis is to run the experiments with one relation, increasing the number of tuples with each experiment and observing the change in precision and recall.

The recall results do not seem particularly striking, though these experiments do not include pattern generalization (other than what a dependency parse provides) or coreference, use a small corpus, and poor candidate-instance matching. Further, as noted above there were other idiosyncrasies in the evaluation that make them only useful for relative comparison, not as general results.

Many of the patterns induced, especially for the acted-in-movie relation, were highly lexical, using e.g. parenthesis or other punctuation to signal the relation. For example, a common pattern was `actor-name (movie-name)`, or `movie-name: actor-name`, e.g. "Leonardo DiCaprio (Titanic) was considering accepting the role as Anakin Skywalker," or "Titanic: Leonardo DiCaprio and Kate Blanchett steam up the silver screen against the backdrop of the infamous disaster." Clearly patterns like this rely heavily on the context and typing to work. In general the pattern `?`x `(?`y`)` is not reliable for the actor-in-movie relation unless you know ?x is an actor and ?y is a movie. However, some patterns, like `?`x `appears in the screen epic ?`y is highly indicative of the relation without the types at all - in fact it is so high precision it could be used to infer the types of ?x and ?y if they were not known. This seems to fit extremely well in our larger reading system, in which the pattern itself provides one form of evidence to be combined with others, but was not a part of our evaluation.

One of the most important things to generalize in the patterns we observed was dates. If patterns like, `actor-name appears in the 1994 screen epic movie-name` could have been generalized to `actor-name appears in the date screen epic movie-name`, recall would have been boosted significantly. As it stood in these experiments, everything but the arguments had to match. Similarly, many relations often appear in lists, and our patterns were not able to generalize that away. For example the sentence, "Mark Hamill appeared in Star Wars, Star Wars: The Empire Strikes Back, and Star Wars: The Return of the Jedi," causes three patterns to be induced; in each, one of the movies is replaced by a variable in the pattern and the other two are required to be present. Then of course all this needs to be combined, so that the sentence, "Indiana Jones and the Last Crusade is a 1989 adventure film directed by Steven Spielberg and starring Harrison Ford, Sean Connery, Denholm Elliott and Julian Glover," would generate a pattern that would get the right arguments out of "Titanic is a 1997 epic film directed by James Cameron and starring Leonardo DiCaprio, Kate Winslett, Kathy Bates and Bill Paxon." At the moment the former sentence generates four patterns that require the director and dates to be exactly the same.

Some articles in the corpus were biographies which were rich with relation content but also with pervasive anaphora, name abbreviations, and other coreference manifestations that severely hampered induction and evaluation.

5 Related work

Early work in semi-supervised learning techniques such as co-training and multi-view learning (Blum and Mitchell, 1998) laid much of the ground work for subsequent experiments in bootstrapped learning for various NLP tasks, including named entity detection (Craven et al., 2000; Etzioni et al., 2005) and document classification (Nigam et al., 2006). This work's pattern induction technique also represents a semi-supervised approach, here applied to relation learning, and at face value is similar in motivation to many of the other reported experiments in large scale relation learning (Banko and Etzioni, 2008; Yates and Etzioni, 2009; Carlson et al., 2009; Carlson et al., 2010). However, previous techniques generally rely on a small set of example relation instances and/or patterns, whereas here we explicitly require a larger source of relation instances for pattern induction and training. This allows us to better evaluate the precision of all learned patterns across multiple relation types, as well as improve coverage of the pattern space for any given relation.

Another fundamental aspect of our approach lies in the fact that we attempt to learn many relations simultaneously. Previously, (Whitelaw et al., 2008) found that such a joint learning approach was useful for large-scale named entity detection, and we expect to see this result carry over to the relation extraction task. (Carlson et al., 2010) also describes relation learning in a multi-task learning framework, and attempts to optimize various constraints posited across all relation classes.

Examples of the use of negative evidence for learning the strength of associations between learned patterns and relation classes as proposed here has not been reported in prior work to our knowledge. A number of multi-class learning techniques require negative examples in order to properly learn discriminative features of positive class instances. To address this requirement, a number of approaches have been suggested in the literature for selection or generation of negative class instances. For example, sampling from the positive instances of other classes, randomly perturbing known positive instances, or breaking known semantic constraints of the positive class (e.g. positing multiple state capitols for the same state). With this work, we treat our existing RDF store as an oracle, and assume it is sufficiently comprehensive that it allows estimation of negative evidence for all target relation classes simultaneously.

The first (induction) phase of LSRD is very similar to PORE (Wang et al., 2007) (Dolby et al., 2009; Gabrilovich and Markovitch, 2007) and (Nguyen et al., 2007), in which positive examples were extracted from Wikipedia infoboxes. These also bear striking similarity to (Agichtein and Gravano, 2000), and all suffer from a significantly smaller number of seed examples. Indeed, its not using a database of specific tuples that distinguishes LSRD, but that it uses so many; the scale of the induction in LSRD is designed to capture far less frequent patterns by using significantly more seeds

In (Ramakrishnan et al., 2006) the same intuition is captured that knowledge of the structure of a database should be employed when trying to interpret text, though again the three basic hypotheses of LSRD are not supported.

In (Huang et al., 2004), a similar phenomenon to what we observed with the *acted-in-movie* relation was reported in which the chances of a protein interaction relation being expressed in a sentence are already quite high if two proteins are mentioned in that sentence.

6 Conclusion

We have presented an approach for Large Scale Relation Detection (LSRD) that is intended to be used within a machine reading system as a source of hypothetical interpretations of input sentences in natural language. The interpretations produced are semantic relations between named arguments in the sentences, and they are produced by using a large knowledge source to generate many possible patterns for expressing the relations known by that source.

We have specifically targeted the technique at the problem that the frequency of patterns occurring in text that express a particular relation has a *very long tail* (see Figure 1), and without enough seed examples the extremely infrequent expressions of the relation will never be found and learned. Further, we do not commit to any learning strategy at this stage of processing, rather we simply produce counts, for each relation, of how often a particular pattern produces tuples that are in that relation, and how often it doesn't. These counts are simply used as evidence for different possible interpretations, which can be supported or refuted by other components in the reading system, such as type detection.

We presented some very early results which while promising are not conclusive. There were many idiosyncrasies in the evaluation that made the results meaningful only with respect to other experiments that were evaluated the same way. In addition, the evaluation was done at a component level, as if the technique were a traditional relation extraction component, which ignores one of its primary differentiators–that it produces sets of hypothetical interpretations. Instead, the evaluation was done only on the top hypothesis independent of other evidence.

Despite these problems, the intuitions behind LSRD still seem to us valid, and we are investing in a truly large scale implementation that will overcome the problems discussed here and can provide more valid evidence to support or refute the hypotheses LSRD is based on:

1. A large number of examples can account for the long tail in relation expression;
2. Producing sets of hypothetical interpretations of the sentence, to be supported or refuted by further reading, works better than producing one;
3. Using existing, large, linked-data knowledge-bases as oracles can be effective in relation detection.

References

[Agichtein and Gravano2000] E. Agichtein and L. Gravano. 2000. Snowball: extracting relations from large plain-text collections. In *Proceedings of the 5th ACM Conference on Digital Libraries*, pages 85–94, San Antonio, Texas, United States, June. ACM.

[Banko and Etzioni2008] Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

[Blum and Mitchell1998] A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 1998 Conference on Computational Learning Theory*.

[Carlson et al.2009] A. Carlson, J. Betteridge, E. R. Hruschka Jr., and T. M. Mitchell. 2009. Coupling semi-supervised learning of categories and relations. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-supervised Learning for Natural Language Processing*.

[Carlson et al.2010] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr., and T. M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*.

[Craven et al.2000] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Sean Slattery. 2000. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, 118(1–2):69–113.

[Dolby et al.2009] Julian Dolby, Achille Fokoue, Aditya Kalyanpur, Edith Schonberg, and Kavitha Srinivas. 2009. Extracting enterprise vocabularies using linked open data. In *Proceedings of the 8th International Semantic Web Conference*.

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

[Gabrilovich and Markovitch2007] Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*.

[Huang et al.2004] Minlie Huang, Xiaoyan Zhu, Yu Hao, Donald G. Payan, Kunbin Qu, and Ming Li. 2004. Discovering patterns to extract protein-protein interactions from full texts. *Bioinformatics*, 20(18).

[Michelson and Knoblock2007] Matthew Michelson and Craig A. Knoblock. 2007. Mining heterogeneous transformations for record linkage. In *Proceedings of the 6th International Workshop on Information Integration on the Web*, pages 68–73.

[Nguyen et al.2007] Dat P. Nguyen, Yutaka Matsuo, , and Mitsuru Ishizuka. 2007. Exploiting syntactic and semantic information for relation extraction from wikipedia. In *IJCAI*.

[Nigam et al.2006] K. Nigam, A. McCallum, , and T. Mitchell, 2006. *Semi-Supervised Learning*, chapter Semi-Supervised Text Classification Using EM. MIT Press.

[Pantel and Pennacchiotti2006] Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st international Conference on Computational Linguistics and the 44th Annual Meeting of the Association For Computational Linguistics*, Sydney, Australia, July.

[Ramakrishnan et al.2006] Cartic Ramakrishnan, Krys J. Kochut, and Amit P. Sheth. 2006. A framework for schema-driven relationship discovery from unstructured text. In *ISWC*.

[Wang et al.2007] Gang Wang, Yong Yu, and Haiping Zhu. 2007. PORE: Positive-only relation extraction from wikipedia text. In *ISWC*.

[Whitelaw et al.2008] C. Whitelaw, A. Kehlenbeck, N. Petrovic, , and L. Ungar. 2008. Web-scale named entity recognition. In *Proceeding of the 17th ACM Conference on information and Knowledge Management*, pages 123–132, Napa Valley, California, USA, October. ACM.

[Yates and Etzioni2009] Alexander Yates and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Artificial Intelligence*, 34:255–296.