# Unsupervised Entity-Relation Analysis in IBM Watson

**Aditya Kalyanpur** ADITYAKAL@US.IBM.COM
**J. William Murdock** MURDOCKJ@US.IBM.COM
IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598

## Abstract

Text paraphrasing algorithms play a fundamental role in several NLP applications such as automated question answering (QA), summarization and machine translation. We propose a novel paraphrasing approach based on an entity-relation (ER) analysis of text. The algorithm uses a combination of deep linguistic analysis (part of speech, dependency parse information) and background resources (NGram, PRISMATIC KB, domain dictionaries) to detect and match entities and relations. We evaluate the ER approach in a QA setting by adding it to the suite of passage scoring algorithms in IBM Watson, a state-of-the-art question answering system. We show a statistically significant improvement in the ability of IBM Watson to identify justifying passages.

## 1. Introduction

Recently, the field of textual paraphrasing and entailment has received a lot of interest (Androutsopoulos & Malakasiotis, 2010). Paraphrasing methods play a crucial role in several NLP applications such as automated question-answering (both, in scoring answers and in providing justifying passage evidence), text summarization, document clustering and machine translation.

These algorithms are typically given two pieces of text (or a question and a passage in the QA setting), and they output the extent to which the two text fragments mean the same thing (or whether the passage entails the question). Several paraphrasing algorithms have been described in literature, including approaches based on bag-of-words (BOW) representation, vector-similarity computations, kernel-based methods (Moschitti et al, 2007), machine-translation inspired techniques (Finch et al, 2005) and logical/semantic form based analysis (Murdock, 2011).

In this paper, we describe a novel text paraphrasing algorithm based on an entity-relation analysis (ER) of the text. The idea behind ER analysis is to detect noun-centric-phrases in the text, each of which corresponds to a singleton entity / concept / instance, and verb-centric-phrases which correspond to relations between these entities.

A key point of this approach is that the entities and relations do not need to be typed, i.e., no pre-existing ontology or taxonomy is required to specify the types and relations of interest. Instead, the algorithm relies mainly on a combination of part-of-speech and dependency-parse information (both of which are provided by standard language parsers), along with background statistical information such as from an NGram KB of the corpus, to detect untyped entities and relations. With that said, however, the algorithm can also benefit from pre-existing dictionaries

© 2015 Cognitive Systems Foundation. All rights reserved.

of known entities and relations to provide insights into what combinations of words form coherent meaningful units. This is a key contribution of this work: the ER scorer can benefit from structured knowledge of entities and relations when they exist but it is not helpless when dealing with instances of texts that are outside the scope of its structured knowledge.

The approach is based on a combination of linguistic rules and background knowledge. It does not require statistical training data. This makes it easier to deploy in new domains: you do not need labeled instances of correct and incorrect paraphrases to make this scorer work. In the event that domain-specific ontologies/type-systems are available, or training data for entity/relation detection is provided, the algorithm can incorporate this knowledge to do better analysis. In addition, if you do have statistical training data, you can train a model that combines this algorithm with other passage scoring algorithms, as described in the evaluation section of this paper.

In this paper, we describe how the ER approach is used for scoring text passages in the IBM Watson QA system (Murdock et al, 2012) though the underlying concepts are applicable in other paraphrasing applications as well.

## 2. Related Work

An in-depth survey of textual paraphrasing and entailment methods is provided in (Androutsopoulos & Malakasiotis, 2010). The IBM Watson QA system also employs a diverse suite of text similarity checking algorithms, which have been described in (Murdock et al, 2012).

The algorithms discussed in literature broadly fall into three main categories:

- Surface String similarity – compute word overlap (often taking word stems or lemma forms to normalize tokens) between the two pieces of text, but ignore grammatical relationships and word order. (e.g. Bag-of-words scorers)
- Syntactic similarity – construct syntactic parses of the two pieces of text and check similarity between the parse trees (e.g. dependency tree edit distance)
- Symbolic meaning representations – construct semantic/logical representations of the two pieces of text and determine equivalence/entailment between the meaning representations (e.g. FrameNet based)

While surface-form based and syntactic similarity techniques have broad coverage, they can be easily confused by superficial similarity. On the other hand, symbolic meaning techniques can perform much deeper analysis of the text, but tend to be a lot more brittle, breaking down on cases that the system has not been trained on or seen before. Much of this brittleness comes from the challenge of taking natural language inputs and converting them into formal knowledge; this can be done (Wang et al., 2012) but only with a modest degree of precision and recall and logical deduction generally requires that inputs be entirely correct. Furthermore, curated knowledge bases can provide very comprehensive and reliable information for a narrow task in a specific domain but generally do not have extensive coverage over broad, open-ended domains.

There also exist machine-learning (ML) based systems, such as the IBM Watson QA system, which combine the various types of text similarity checking techniques mentioned above into a single model. This is done by transforming the various algorithm outputs into feature scores in the model, and using pre-compiled training data to learn how to appropriately weigh and balance the

different features/approaches. Through experiments, such ML systems typically outperform individual techniques (Murdock et al, 2012). Still, there remains the potential to add new and useful, complementary approaches into such a machine learning based system, and demonstrate impact over and above the existing techniques.

The proposed ER-based technique is a novel hybrid between syntactic similarity measures and measures operating on symbolic meaning representations. While the approach uses dependency parse information for text matching, it operates on the notion of entities and relations in a semantic graph. The resultant solution provides a compromise between these extremes, gaining some of the breadth of syntactic methods and some of the depth of symbolic meaning methods; a point reinforced by our evaluation (Section 5). The evaluation also shows the additional impact of the ER-based scorer in an ML system that previously contains a large set of state-of-the-art text similarity techniques.

## 3. Motivating Example

In this section, we describe how the ER analysis is used in an automated QA setting to compare a question and a passage.

Table 1 shows a sample question and passage taken from the medical domain. The underlined phrase in the passage, "Kidney Amyloidosis", is the candidate answer that is being scored. The passage provides a strong justification that this answer is correct, especially if it is known that "proteinuria" is the medical term for abnormal urine albumin and that Kidney Amyloidosis is a disease.

The ER algorithm starts by detecting entities and relations in the question and passage. Entities are not necessarily contiguous terms (e.g. QE2, QE3). Detection involves merging noun phrases to create more coherent entities (e.g. merging "abnormal quantities", "urine albumin"). The entity detection algorithm also associates a confidence score with each entity (not shown above), that is used in the final score computation.

**Question**: "*What kidney disease causes swelling around wrist and knee joints and gradually leads to abnormal quantities of urine albumin?*"

**Passage**: "*Kidney Amyloidosis manifests as peripheral joint swelling, cysts in bones, and proteinuria"*

**Entities detected in Question:**
QE1: *kidney disease*
QE2: *swelling around wrist joint*
QE3: *swelling around knee joint*
QE4: *abnormal quantities of urine albumin*

**Entities detected in Passage:**
PE1: *Kidney Amyloidosis*
PE2: *peripheral joint swelling*
PE3: *cysts in bones*

| |
|---|
| PE4: *proteinuria*<br><br>**Relations detected in Question:**<br>QR1: *QE1 – causes – QE2*<br>QR2: *QE1– causes – QE3*<br>QR3: *QE1– gradually leads to – QE4*<br><br>**Relations detected in Passage:**<br>PR1: *PE1 – manifests as – PE2*<br>PR2: *PE1– manifests as – PE3*<br>PR3: *PE1– manifests as – PE4*<br><br>**Entity Matches Found:**<br>QE1 – PE1; QE2 – PE2; QE3 – PE2; QE4 – PE4<br><br>**Relation Matches Found:**<br>QR1 – PR1; QR2 – PR1; QR3 – PR3 |

**Table 1: Motivating Example of ER Analysis**

Relation detection follows entity detection and uses heuristics based on POS and parse information (see next section) to detect 'relation-bearing' phrases between a pair of entities. As in the case of entities, relations are also associated with a detection confidence score.

After the detection phase, the ER algorithm attempts to match the entities and relations found between the question and the passage. The information to do entity matching comes from a variety of techniques such as a statistical (e.g. distributional similarity between "wrist joint" and "peripheral joint" from a large medical corpus), or a knowledge based method (e.g. definition of "proteinuria" from a medical KB). The entity match scores reflect the confidence of the matching algorithm.

Note that the algorithm only matches relations when their corresponding entity arguments also match. As in the entity matching case, a variety of techniques (e.g. distributional similarity, thesaurus-lookup etc) are used to match relation phrases, and a score between 0 and 1 is assigned to each relation-pair match.

The final score computed by the ER algorithm is a combination of the entity and relation detection and matching scores. The algorithm creates a "question graph" (where entities are nodes and relations are edges) and a corresponding "passage graph." It then computes the extent to which both the question and passage graphs align/overlap, considering the matching nodes and edges. In our example, the algorithm finds many relation matches between question and passage relation pairs, resulting in a strong overlap between the question and passage graphs, and thus outputs a high similarity score.

## 4. ER Algorithm

In this section, we briefly summarize the four main modules of the algorithm. None of these steps require statistical training; they involve a combination of rules and background knowledge sources.

### 4.1 Entity Detection Module

The entity detection module uses linguistic characteristics to generate a large set of candidate entities and then uses background knowledge to provide evidence for whether those candidate entities are semantically important. For example, consider the noun "pain" in the sentence "The patient reported remarkably severe pain in the knee last week." We can think of "pain" as an entity by itself or in combination with any or all of the words or phrases that modify it, e.g., "severe pain", "remarkably severe pain", "pain in the knee", "pain in the knee last week", etc. However, some of these entities are more important than others semantically; for example, a medical lexicon may list "pain in the knee" as an entity but is unlikely to list "pain in the knee last week" as an entity. Syntactic structure can identify combinations of words that *could* be an important entity, and background knowledge can help determine which ones actually are.

**Entity Detection**
**Input:** Text (e.g. sentence, passage, document)
**Output:** Entities found in the text, each associated with a score reflecting the detection confidence
**Algorithm:**

1. **Detect noun phrases** found by a dependency parser (McCord et al, 2012)
   - Consider denominal verbs and adjectives when forming noun-centric phrases
2. **Generate potential entities as follows:**
   - Merge adjacent / contiguous nouns in the text (e.g. "urine albumin")
   - Use noun-adjective syntactic modifier relationships in the parse analysis to attach adjectival modifiers when the head noun is common (e.g. "knee joint"), where commonality is determined by corpus frequency statistics
   - Use noun-preposition syntactic modifier relationships in the parse analysis to attach prepositional phrases when the head noun is common (e.g., "swelling around wrist joint"), again using corpus frequency
   - Use domain-specific knowledge about syntactic and semantic representation of entities to detect potential entity spans. For example, in the medical domain, *symptoms* are characterized by attributes such as *body-part*, *severity*, *onset* etc. This information can be encoded as rules to formulate complete symptom spans using part-of-speech and semantic type information on neighboring words.
3. **Compute support for potential entities using several background resources:**
   - Standard N-gram corpus of the domain – check frequency of entity mentions
   - PRISMATIC (Fan et al, 2012) KB – check frequency of syntactic parse structures.
   - Domain dictionary – check if entity mention is a concept in the dictionary

The output of this step is a background support vector, where each dimension is the support value from each of the different knowledge bases

4. **Final Score Computation:** Use a rule-based or statistical model to transform the background support vector into a final confidence score for each entity (essentially this means determining appropriate thresholds/weights for support)

**Table 2: Entity Detection Algorithm**

Some notes on the module:

- Step 3 can employ a wide variety of resources including statistical information about word sequences (an NGRAM index or similar), a knowledge-base extracted from a large body of text (such as PRISMATIC) or a curated dictionary (such as UMLS).While both an NGRAM index and PRISMATIC can be used to extract statistical information (frequency, PMI etc) about words and phrases in the corpus, the key difference is that NGRAMS operate on the surface form of the text, while PRISMATIC operates on parse information, and hence can capture longer distance dependencies between words. For example, given the entity: "*pain in the knee*", which can be expressed syntactically as: *<noun:"pain", prep:"in", obj-prep:"knee">*, we can obtain frequency counts of this particular syntactic structure in the corpus from PRISMATIC. For the corpus sentence: "*The patient experienced pain and discomfort in the hands, legs and knees*", a 4-gram index will not find a match for the entity, but PRISMATIC would.

- The final step involves determining suitable thresholds for background support, which can be done using some default metrics (e.g. frequency>some reasonably large number), or by leveraging examples of domain entities (if available) in order to train a model to learn the thresholds. Most domains have dictionaries or glossaries that can be used to bootstrap this process. For example, in the medical domain, we use the UMLS Meta-thesaurus (http://www.nlm.nih.gov/pubs/factsheets/umlsmeta.html).

### 4.2 Relation Detection Module

The relation detection module finds combinations of terms that collectively indicate how entities are connected to each other. As with entity detection, relation detection uses syntactic structure to identify relations, however it drops the notion of background support as we assume there is a much wider variety in relation expression.

**Relation Detection**
**Input:** Pair of Entities
**Output:** Relations between input entities (if any)
**Algorithm:**

1. **Find path linking the input entities in the dependency parse of the text**
   - Limit path length to a certain value, e.g. 10 elements, where an element is a node or edge
   - Return no relation if no path found within specified distance limit
2. **Check that the path is a *valid* relational path as follows:**
   - Case 1: If the dependency path is bidirectional: Let the path be {P1, P2} where P1 is sub-path from one entity to the root (the point at which the path changes direction) and P2 is the sub-path from the root to the other entity.
     - Return *valid* if P1 contains a verb-subject relation and P2 contains an verb-object or verb-predicate relation (or vice versa, in which case the entity arguments are reversed)
     - Return *valid* if P1 contains an verb-object relation and P2 starts with a any adverbial construct (e.g., adverb, adverbial prepositional phrase)

| |
|---|
| modifying a verb.<br>• Case 2: If the path is unidirectional: return *valid* if there are no conjuncts (e.g., and, or) in the path (i.e. the two entities are not part of the same conjunction)<br>**3.** If valid relational path found in step (2), **generate the complete relational phrase** by adding adverbs and other modifiers in surrounding text (e.g. "..gradually leads to...")<br>**4.** **Final Score computation:** Compute a confidence score for the relation considering the length of the path (shorter the path, higher the confidence) and which validity checking heuristic fired (case 1 has higher confidence than case 2) |

**Table 3: Relation Detection Algorithm**

Some notes on the module:

- The underlying intuition of step (2) is that relation-bearing phrases typically involve verbs or predicates, and thus the dependency path linking the entities must involve some verb-centric dependency relations such as subject, object, predicate, and adverbial modifiers (which are recognized by dependency parsers).
- Within step (2), case 2 applies when the phrase is a prepositional phrase that wasn't deemed to be part of an entity span – e.g., "pain in the knee *over the* weekend", where the underlined phrases represent entities found by the entity detection module and the italicized phrase represents the relational phrase (in this case, it has a temporal semantic)

### 4.3 Matching Modules

The algorithm employs two matching modules:

- **Entity matching module** – which determines the extent to which entities appear to be equivalent or opposites
- **Relation matching module** – which determines the extent to which connections between entities appear to be equivalent or opposites

Both modules use a combination of word and phrase matchers to determine similarity between sets of terms; in the former case, it is the terms that comprise the entity span, while in the latter, it is the terms that make up the relation expression.

Examples of term matchers are shown in the table below:

| Matching Approach | Examples |
|---|---|
| *String / Lexical* | Levenstein distance between terms; Word overlap between phrases |
| *Dictionary/ Thesaurus* | WordNet distance between terms; Abbreviation-expansions from a glossary |
| *Distributional Similarity* | LSA similarity between terms; PMI from a corpus index |
| *Semantic Types* | Check if one term is an instance/type of the other |

**Table 4: Term Matching Approaches**

Note that each of the term matchers above produces a score between -1 to 1 (where -1 indicates opposites, 0 indicates no match, and 1 indicates synonyms/equivalent terms). The current version of the algorithm ignores scores less than 0, though we are working on extending the framework to handle mismatch information. Also, we have experimented with various strategies for merging the various term-matching scores, and in our experience, the default of Max absolute score works well in most cases.

### 4.4 Final Scoring

Entity detection and relation detection produce a graph of entities and relations for both the question and the passage. Entity matching and relation matching determine how well the nodes and edges of the question graph match those of the passage graph. Final scoring combines the results of entity matching and relation matching to draw a final conclusion about how well the entities and relations of the question align with the entities and relations of the passage.

The algorithm used to match the graph is similar to the one used in LFACS (Murdock, 2011), which is based on the Structure Mapping algorithm of the Structure Mapping Engine (Falkenhainer et al., 1989). The main difference between LFACS and the ER algorithm described in this paper is that LFACS operates at the level of individual lexical elements (nouns, adjectives, verbs, etc.). In contrast, the ER algorithm operates at a higher level of abstraction: entities and relations that may consist of multiple syntactic units and whose semantic significance is validated by background knowledge.

## 5. Evaluation

In order to compare the ER paraphrasing technique to other state-of-the-art paraphrasing methods, we manually created a benchmark dataset. The data was taken from the medical domain, as that is our current domain of interest.

Two annotators manually inspected triplets of the form: <question, candidate answer, answer-bearing-passage> and provided a binary judgment of whether the passage provides a justification for the candidate answer being the correct answer to the question. Partial justifications (e.g. when the passage provided evidence for only a part of the question) were treated the same as full justifications.

As noted earlier, the ER scorer does not require training data since it is based on rules and background knowledge. However, IBM Watson's method for integrating multiple distinct scoring features (Gondek et al., 2012) does depend on statistical training data.

The instances (triplets) were obtained using a set of training questions, each of which was of a multiple choice format with five answer choices (and only one correct answer), and a medical corpus which included resources such as DynaMed (2015), Elsevier and Wikipedia. Relevant passages from the corpus were found by issuing queries using the Indri search engine (Strohman et al, 2005); query terms included the candidate answer (compulsory) and all keywords from the question (optional). The generation and execution of the query is completely automated and uses the same core mechanisms used in the IBM Watson system to find evidence for scoring answers to questions.

Only instances with inter-annotator agreement were selected in the final dataset as our ground truth (Kappa score was 0.8). In all, the ground truth contained 1200 instances, of which 86% were negative examples, reflecting the fact that a large proportion of passages to a given question are not justifying.

We then used this data to train and test a logistic classifier over a set of paraphrasing features. The train and test set sizes were 800 and 400 instances respectively (randomly sampled from the full ground truth).

For evaluation purposes, we trained two separate model configurations – C1: a baseline model using the full suite of paraphrasing (also called passage scoring) algorithms in IBM Watson, a state-of-the-art QA system (Murdock et al, 2012), and C2: a new model in which we added the ER feature to the baseline feature set. The results are shown in Table 5.

| System | AuC | Pearson's R | Correct Ratio | Incorrect Ratio |
|---|---|---|---|---|
| **C1: Baseline** | .618 | 0.603 | 2.93 | 0.696 |
| **C2: Baseline + ER** | .715 | 0.649 | 3.36 | 0.616 |

**Table 5: Impact of ER Passage Scoring on Complete System**

Since this is a binary classification task, we report results for Area under the Precision-Recall curve (AuC) and correlation with correctness (Pearson's R). As shown, adding the ER scorer to the baseline results in improvement across all three metrics. We assessed statistical significance using Fisher's randomization test and found that the improvement in AuC and Person's R were statistically significant ($p=0.01$ and $p=0.04$, respectively). As the baseline IBM Watson system already uses 20 state-of-the-art paraphrasing/passage scoring features, strong across-the-board gains for this crucial text-analytic task implies that the ER scorer is providing considerable new value.

We also report results for two less common metrics that may provide additional insights into the behavior of the system. The "Correct Ratio" is the mean score on passages labeled as "correct" (i.e., that the passage does justify the answer) divided by the mean score on all passages. The "Incorrect Ratio" is the mean score on passages labeled "Incorrect" divided by the mean score on all passages. A system with a high correct ratio and a low incorrect ratio is particularly effective in the degree to which it awards higher overall confidence scores to correct answers than to incorrect answers. This trait is particularly important in applications where the degree of confidence is exposed to an end user either explicitly (e.g., showing a numerical confidence or a slider) or implicitly through behavior (e.g., only showing high confidence

We also looked at how the ER scorer alone compares to each of the existing passage scoring features in IBM Watson and how it compares to the combination of all together. In both AuC and Pearson's R, the ER scorer outperforms each of the other scorers in IBM Watson but still falls well short of all of them combined. In Table 6, we show how the ER scorer compares to two other scores: LFACS (Murdock, 2011) and Passage Term Match (Murdock et al., 2012).

| System | AuC | Pearson's R | Correct Ratio | Incorrect Ratio |
|---|---|---|---|---|
| **PassageTermMatch** | .434 | .451 | 1.92 | 0.850 |
| **LFACS** | .300 | .289 | 3.71 | 0.559 |
| **ER** | .540 | .523 | 3.89 | 0.553 |

**Table 6: Comparison of Passage Scoring Components**

As noted earlier, LFACS uses Structure Mapping to align a graph of the question terms to a graph of passage terms; the ER scorer uses a similar alignment strategy but at a higher level of abstraction: entities and relations. In contrast, Passage Term Match is a very simple bag-of-words matcher that completely ignores the structure of the text and the order of the words. Table 6 shows that Passage Term Match dramatically outperforms LFACS in AuC and Pearson's R; this reflects the fact that LFACS is very brittle and thus falls far short of Passage Term Match in terms of overall coverage. The brittleness of LFACS is a result of the extremely fine granularity of the graphs it uses; to get a high LFACS score, a passage must use terms that match many question terms and have those terms connected to each other in precisely the same way (either by the dependency parser or by semantic relation detectors that use a rigid pre-defined ontology). These requirements limit the applicability of LFACS, but they also ensure that a particularly strong score for LFACS is a very strong indicator of a passage being correct. This is reflected by the "correct ratio" and "incorrect ratio" scores for LFACS, which shows that it does much better than Passage Term Match in the degree to which it prefers right answers over wrong ones. A passage for which Passage Term Match has a very high score matches most or all of the question terms in any order and in any configuration; that alone is not enough to be highly confident that the passage actually justifies the answer.

The ER scorer is designed to combine the strengths of Passage Term Match (broad applicability) with the strengths of LFACS (ability to be highly confident in conclusions). Table 6 shows that it does succeed remarkably well at combining these strengths. It provides comparable (slightly better) correct and incorrect ratios, reflecting the fact that, like LFACS, it is verifying that not only are the concepts in the question present in the passage but also that they relationships among those concepts are aligned. However, because the entities and relations in the ER scorer are at a higher level of abstraction, it can match them using less brittle, more broadly applicable methods (as described in Sections 4.3 and 4.4). As a result, it dramatically surpasses both LFACS and Passage Term Match in AuC and Pearson's R. In fact, none of the other passage scorers in Watson exceed 0.5 for either of these metrics (putting the ER scorer's 0.540 and 0.523 comfortably in first place). However, the full system (as shown in Table 5) does substantially better than the ER scorer alone. Thus while the ER scorer outperforms any one of the other scorers in Watson it is more effective as a complement to the other scorers than it would be replacing all of them.

## 6. Conclusion and Next Steps

In this paper, we describe a novel unsupervised text analytic technique that is based on the idea of identifying untyped entities and relations in the text. Results based on the ER approach provide a significant performance improvement over a baseline state-of-the-art passage scoring system (IBM Watson) on a justifying passage task, which is a core task in automated QA for both, scoring answers and providing evidence. We also demonstrate via our evaluation how the ER approach combines the best aspects of syntactic similarity measures (breadth/recall) with semantic representations based techniques (depth/precision). We believe that the ER text analytic can play a crucial role in natural language processing applications such as QA, paraphrasing, summarization and language translation.

There are two extremes that often arise among systems that perform cognitive tasks:

- Systems based on statistical correlations often perform reasonably well but they are inherently limited by the lack of deeper understanding. We can build a system that does

a good job extracting the statistical signal that is available in a large quantity of data, and it works fairly well. However, once we have that system it is hard to make it any better because there is only so much signal to be pulled from the data; progress asymptotically approaches a limit that is still well short of acceptable for many applications.

- Systems based on knowledge and deduction are virtually limitless in theory; when we encounter a particularly challenging example we can always add new background knowledge and/or inference rules to address that example. However, complex open domains generally provide a more diverse variety of requirements than a knowledge-base engineer can cover. Reasoning over knowledge provides very precise, reliable conclusions when it is applicable but frequently it does not apply at all.

In general, IBM Watson addresses this conundrum using an "all of the above" strategy in which a variety of statistical and knowledge-based subcomponents are all applied to all inputs and conventional machine learning methods are used to integrate these subcomponents. This compromise allows IBM Watson to produce correct answers with very high confidence when its more deductive components are applicable while still being able to produce an informed guess when they are not. However, when all the subcomponents lie on either extreme they are collectively still limiting because it is prohibitively expensive to get very broad coverage from the very precise components. Thus highly confident answers backed by deduction are very infrequent and a large proportion of the system outputs are driven by broadly applicable but imprecise correlations. Our ER scorer addresses this challenge by not going to either extreme. It is not limited to only considering types of entities and relations for which it has deep semantic knowledge. However, it is able to benefit from background knowledge and it does reason at a higher level of abstraction than simple co-occurrence of words. The ER scorer represents just one point on a broad spectrum of capabilities that are neither extremely deep nor extremely shallow. Our results with it suggest that it is a valuable contributor to a broad collection of scorers.

We continue to explore improvements to the ER approach, namely:

- Support for multiple distinct interpretations of the text (the same phrase can act as an entity or relation depending on neighboring interpretations)
- Support for n-ary relations (currently all relations are binary)
- Considering negated entity and relation spans, along with opposite/antonymy information, to produce a text mismatch score

## References

Androutsopoulos, I., Malakasiotis, P. (2010). A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*. Volume 38 Issue 1, Pages 135-187.

Dynamed. (2015). DynaMed Overview. Available: https://dynamed.ebscohost.com/about/about-us

Falkenhainer, B., Forbus, K.D. and Gentner, D. (1989). The Structure Mapping Engine: Algorithm and examples. Artificial Intelligence, 41, 1-63.

Fan, J., Kalyanpur, A., Gondek, D., and Ferrucci, D. (2012). Automatic Knowledge Extraction from Documents. *IBM Journal of Research and Development,* Volume:56, Issue: 3.4.

Finch, A., Hwang, Y-S, and Sumita, E. (2005). Using machine translation evaluation techniques to determine sentence-level semantic equivalence. *In Proc. of the 3rd Int. Workshop on Paraphrasing*, pp. 17–24, Jeju Island, Korea.

Gondek, D.C., Lally, A., Kalyanpur, A., Murdock, J.W., Duboue, P.A., Zhang, L., Pan, Y., Qiu, Z.M., and Welty, C.. (2012). *IBM Journal of Research and Development*, Volume:56, Issue: 3.4.

Moschitti, I., Quarteroni, S., Basili, R., and Manandhar, S.. (2007). Exploiting syntactic and shallow semantic kernels for question/answer classification. *In Proc. 45th ACL Conf.*, Prague, Czech Republic. Available: http://www.ist-luna.eu/pdf/ACL07.pdf.

Murdock, J.W., Fan, J., Lally, A., Shima, H. and Boguraev, B. (2012). Textual evidence gathering and analysis. *IBM Journal of Research and Development*, Volume:56, Issue: 3.4.

Murdock, J.W. (2011). Structure Mapping for Jeopardy! Clues. *International Conference on Case-Based Reasoning, ICCBR 2011*, London, UK, September 12-15, 2011. pp 6-10

McCord, M., Murdock, J.W., and Boguraev, B. (2012). Deep parsing in Watson. *IBM Journal of Research and Development*, Volume:56, Issue: 3.4.

Strohman, T., Metzler, D., Turtle, H., Croft, W.B. (2005). Indri: a language-model based search engine for complex queries. *Proceedings of the International Conference on Intelligent Analysis*, 2005

Wang, C., Kalyanpur, A., Fan, J., Boguraev, B.K., Gondek, D.C. (2012). *IBM Journal of Research and Development,* Volume:56, Issue