Evaluating Relation Inference via Question Answering Supplementary Material

Omer Levy **Ido Dagan**
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
{omerlevy,dagan}@cs.biu.ac.il

Extracting Open IE Assertions

We extracted over 63 million unique subject-relation-object triplets from Google's Syntactic N-grams (Goldberg and Orwant, 2013), including the number of times each one appeared in the corpus. This collection represents over 1.5 billion distinct appearances. The assertions may include multi-word phrases as relations or arguments, for example:

$$(\text{chocolate}, \text{is made from}, \text{the cocoa bean}) : 22$$

We extracted these triplets using the following method:

1. Obtain the collection of biarcs, triarcs, and quadarcs (the *extended* versions). The *extended* collections contain function words such as modality and negation.

2. Select all syntactic n-grams that contain a subject (including passive and controlling), followed by a verb at the n-gram's root, and ending with an object (direct, prepositional, or copular). Other tokens may appear in between these ones and before the subject.

3. Retain all n-grams that have exactly one subject and one root.

4. Define the subject as the subtree rooted at the subject token.

5. Define the object as the subtree rooted at the last token in the n-gram.

6. Verify that the noun-phrases describing subjects or objects are continuous, and contain only internal dependencies of the types: det, $amod$, nn, or dep.

7. Verify that the tokens between the subject noun-phrase and object noun-phrase form a subtree that includes the root. This span is the assertion's relation.

8. Yield assertions of the form $(subject, relation, object) : count$, where each element is a string, discarding syntactic information.

9. Aggregate the frequencies of identical assertions.

10. Remove assertions whose relations appear with less than 10 unique subject-object pairs.

Retrieving Answer Candidates

We defined three criteria for matching an answer candidate to a question. We now translate them into a process for retrieving answer candidates for a given question q. Some of the filtering steps below are applied to increase the diversity of answers and reduce the amount of non-grammatical assertions.

1. Retrieve all assertions where one of the arguments (subject or object) is equal to q_{arg}, ignoring stopwords and inflections. The matching argument is named a_{arg}, while the other (non-matching) argument is named a_{answer}.

2. For each retrieved assertion, retain those where a_{answer} is a type of q_{type}. We use WordNet 3.0 (Fellbaum, 1998) as our taxonomy T. To determine the sense of q_{type}, the first WordNet sense was selected. For a_{answer}, we used Lesk (Lesk, 1986) via NLTK (Bird et al., 2009) to choose the appropriate sense. If q_{type} can be reached from a_{answer} using only hypernym or instance hypernym edges (including combinations of both), then the assertion is retained.

Assertions in which a WordNet sense was not found for a_{answer} were discarded.

3. Retain assertions that contain only *maximal* relations and arguments; i.e. if some relation x_{rel} contains y_{rel} as a subsequence, the assertion y that contains y_{rel} is discarded. A similar rule is applied to a_{answer}, but not to a_{arg}. This phase significantly reduces the number of candidates that have missing content words. For example, the assertion $(\text{France}, \text{declared on}, \text{Germany})$ is malformed, because its relation is missing the content word "war". Usually, another assertion with a subsuming relation ("declared war on") will be present, indicating that the shorter predicate should be discarded.

4. Discard all assertions where $a_{rel} = q_{rel}$.

5. Order the remaining assertions by corpus frequency, and remove less-frequent assertions so that no a_{rel} appears more than 3 times and no a_{answer} appears more than 7.

6. Remove duplicate candidate answers.

Improving Annotation Quality

To improve annotation quality, we implemented two mechanisms. As an online mechanism, we injected two hand-crafted candidate answers (one positive and one negative) to roughly 300 of the 1,500 questionnaires (these were written for 120 of the 573 questions). Workers who did not answer these examples correctly could not complete the task, and were asked to go back and fix their annotations.

As an offline mechanism, we tried to ignore workers that consistently gave poor annotations. We measured the amount of times each annotator was in a sharp minority (1/5) with respect to other annotators. We then calculated each annotator's "insurgency" rate as follows:

$$insurgency = \frac{N_{minority}}{N_{examples} + 25}$$

Where $N_{examples}$ is the total number of examples annotated by the worker, and 25 is a smoothing constant. Annotations from workers with over 20% insurgency were discarded.

Embeddings

We create two sets of embeddings in this paper.

Relation Embeddings Similar to DIRT (Lin and Pantel, 2001), we create vector representations for relations, which are then used to measure semantic similarity using cosine similarity. From the set of assertions extracted in Section 3, we create a dataset of relation-argument pairs, each pair containing a single argument. For example, the assertion (chocolate, is made from, the cocoa bean) is split into two relation-argument pairs:

is made from, x: chocolate

is made from, y: the cocoa bean

Notice that we add the argument's position (x for subject, y for object) in each case. Over this dataset, we run word2vecf (Levy and Goldberg, 2014) to train the embeddings, where the relations take the place of the target, and the arguments take the place of the context. We use 500 dimensions, 1 negative sample per positive example, context distribution smoothing ($\alpha = 0.75$), and 10 iterations.

Word Embeddings We used the *extended uniarcs* from Google's Syntactic N-grams (Goldberg and Orwant, 2013) to create a dataset of word-context pairs. We follow Levy and Goldberg's (2014) methodology and extract two instances from each dependency arc, while collapsing prepositional objects. For example, from the syntactic n-gram "game of thrones", we extract:

game, pobj:of↓thrones

thrones, pobj:of↑game

We discard instances containing words and/or contexts that appeared less than 100 times in total. Over this dataset, we run word2vecf (Levy and Goldberg, 2014) to train the embeddings. We use 500 dimensions, 5 negative sample per positive example, context distribution smoothing ($\alpha = 0.75$), and 10 iterations.

References

[Bird et al.2009] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

[Fellbaum1998] Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

[Goldberg and Orwant2013] Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of*

the Main Conference and the Shared Task: Semantic Textual Similarity, pages 241–247, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

[Lesk1986] Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, pages 24–26. ACM.

[Levy and Goldberg2014] Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, June. Association for Computational Linguistics.

[Lin and Pantel2001] Dekang Lin and Patrick Pantel. 2001. Dirt: Discovery of inference rules from text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328. ACM.