The Effectiveness of Traditional and Open Relation Extraction for the Slot Filling Task at TAC 2011

Filipe Mesquita[*], **Ying Xu**[*], **Aditya Bhargava**[†]
Mirko Bronzi[‡], **Denilson Barbosa**[*], **Grzegorz Kondrak**[*]
[*]University of Alberta, [†]University of Toronto, [‡]Roma Tre University
`{mesquita,yx2,denilson,kondrak}@cs.ualberta.ca`

Abstract

Our goal in this paper is to investigate the effectiveness of relation extraction techniques for the slot-filling task. We discuss two relation extraction systems. YRES follows the traditional paradigm in relation extraction, where a system takes advantage of available examples for each relation to be extracted. On the other hand, SONEX follows the open relation extraction paradigm, where the relations to be extracted are assumed to be unknown *a priori*. In particular, SONEX applies clustering techniques to identify relations in an unsupervised way. The results of our submissions show that the performances of the two systems are fairly similar.

1 Introduction

The TAC slot-filling task comprises the identification of values for certain attributes, or *slots*, about an entity. Each named entity is given as a query and the answers are often called *slot fillers*. There are 100 queries this year, half being person and half being organizations. Examples of slots are `per:date_of_birth` (a person's date of birth) and `org:subsidiaries` (a organization's subsidiaries). Each slot allows either a single answer (e.g., `per:date_of_birth`) or a list of answers (e.g., `org:subsidiaries`).

In the past, slot-filling has been attempted with techniques from different fields, such as question answering, machine learning, information retrieval and information extraction. We are particularly interested in investigating the viability of relation extraction techniques for this task. Relation extraction aims at identifying relations among named entities in text. For example, the sentence "Baraka Obama is married to Michelle Obama" presents the relation "married to" between the entities "Baraka Obama" and "Michelle Obama". Every slot is concerned as a relation between the query and the slot value.

We discuss the application of two relation extraction systems to the slot-filling task: SONEX and YRES. YRES follows the traditional paradigm in relation extraction, where the relations of interest are known and examples for each relation are available. YRES extracts *trigger words* for each slot. For example, "parent", "dad", and "mother" are trigger words for the `per:parents` slot. On the other hand, SONEX follows the *open* relation extraction paradigm (Banko and Etzioni, 2008), where the relations of interest are too many (requiring great effort to provide examples) or unknown a piori. Therefore, open relation extraction systems aim at extracting every relation from a collection with no relation-specific training. Both systems got higher precision and lower recall than median.

The remainder of the paper is organized as follows. Section 2 discussed the related work. Section 3 presents the preprocessing of the two systems, such as part of speech tagging and named entity recognition. Section 4 describes the SONEX system and Section 5 describes the YRES system. Section 6 presents our post-processing step. This step includes converting the relations extracted by each system into answers for slots. Section 7 analyzes the results. Finally, Section 8 concludes the paper.

2 Related work

The traditional approach to relation extraction is defined as a classification problem: given a relation

R and a pair of entities in a sentence S, does S asserts R between this pair of entities? Supervised systems use manually labeled examples to train a classifier for each relation. This classifier is either based on extracted features (GuoDong et al., 2005) or kernel functions (Zelenko et al., 2003; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005). Bootstrapping systems require significantly less training data. These systems discover new relation instances by using a small set of entity pairs (Brin, 1998; Agichtein and Gravano, 2000) or hand-crafted extraction patterns (Etzioni et al., 2004). Our system YRES tries to reduce the training effort even further by requiring only a few keywords that indicate a relation.

A limitation of the traditional approach to relation extraction is that it scales linearly with the number of relations. Open relation extraction (ORE) is a new paradigm that aim at overcoming this limitation by extracting *unanticipated* relations (Banko and Etzioni, 2008). ORE systems (Banko and Etzioni, 2008; Zhu et al., 2009; Hasegawa et al., 2004; Mesquita et al., 2010) are designed to extract any relation expressed in text with no relation-specific training. The seminal work of TextRunner (Banko and Etzioni, 2008) proposes a CRF model to recognize tokens describing a relation between a pair of entities. O-CRF relies on relation-independent features, such as prepositions and part-of-speech tags. Our system SONEX (Mesquita et al., 2010) is based on the approach proposed by Hasegawa et al. (2004). In this approach, the relationship between two entities is indicated by the sentences where they appear together. In order to find all pairs of entities belonging to the same relation, a clustering algorithm is used to group pair of entities that are cited in similar sentences.

3 Pre-processing

The first step in our approach is to convert each document in the TAC source collection into plain text for easier manipulation. We start by replacing HTML tags (e.g., <p>) with their corresponding plain text alternatives. Next, we convert Unicode characters to ASCII.

Each document is annotated by the Stanford natural language processing tool[1] as follows. First, this tool splits each document into sentences. A tokenizer identifies individual tokens in a sentence. In addition, the tool annotates each token with part-of-speech tags and lemma.

Next, we identify mentions to entities in a sentence. We use the Stanford named entity recognition system (Finkel et al., 2005) to perform this task. This system is able to recognize several types of entities, such as people, organizations, locations, dates and numbers. In addition, we identify mentions that refer to the same entity using OrthoMatcher (Bontcheva et al., 2002), a coreference resolution system provided by the GATE framework[2].

4 SONEX

SONEX identifies the relationship (if any) between entities e_1, e_2 by analyzing the sentences that mention e_1 and e_2 together. An *entity pair* is defined by two entities e_1 and e_2 together with the *context* in which they co-occur. For our purposes, the context can be any textual feature that allows the identification of the relationship for the given pair. As an illustration, Table 1 shows entity pairs where the context consists of the exact text *in between* the mentioned entities. As we discuss later, we actually employ techniques to *extract* the context from the text in the sentences.

We extract entity pairs as follows. For each sentence, we search for pairs of entities separated by at most five intervening words. We store each entity pair along the sentences where they appear in a Lucene[3] index. This index enables several types of search operators and provides an efficient mechanism (inverted indices) to store statistics about tokens and entity pairs.

SONEX works by clustering entity pairs with similar context. Once this clustering step is done, each cluster is analyzed and a common label is assigned by inspecting the contexts of the pairs within the cluster. The intuition behind this approach is that entity pairs belonging to the same relation often present similar contexts. We use the context of an

[1] `http://nlp.stanford.edu/software/corenlp.shtml`

[2] `http://gate.ac.uk/`

[3] `http://lucene.apache.org/`

Entity 1	**Context**	**Entity 2**
⟨Barack Obama, `PER`⟩	's wife better than his wife and	⟨Michelle Obama, `PER`⟩
⟨Tom Cruise, `PER`⟩	, accompanied by wife 's fiancee , is inspired by	⟨Kate Holmes, `PER`⟩

Table 1: Entity pairs and their context from the TAC source collection.

entity pair to produce a vector where each dimension corresponds to a token in the context. Therefore, every pair is represented by a vector in the Vector SpaceModel (Manning et al., 2008).

Many tokens in the context are irrelevant to describe a relation. These tokens are often considered as noise. This includes prepositions, determiners, adjectives, adverbs, among others. We also observed that relations are often better described by verbs and nouns. Therefore, we consider only those tokens tagged as verb or noun. In order to make no distinction between tokens that are inflections of the same word (e.g., "acquired", "acquires"), we replace the original token by its lemma (e.g., "acquire").

A token t in a context vector is weighted by the widely-adopted $tf \cdot idf$ weighting scheme (Manning et al., 2008). Here, tf is the normalized frequency of the token in the context, while $idf = log(\frac{|D|}{d:t \in d})$, where $|D|$ is the total number of entity pairs,and $d : t \in d$ is the number of entity pairs that contain the token t at least once. We compute the similarity between context vectors by using the cosine similarity measure. We use the Mahout library[4] to store and manipulate context vectors.

4.1 Clustering algorithm

In order to cluster context vectors, we use the Hierarchical Agglomerative Clustering (HAC) algorithm (Manning et al., 2008). One of the advantages of HAC is that it does not require the number of clusters to be produced as a parameter. This is a desirable property since one can seldom estimate the number of relations described in a collection *a priori*. On the other hand, HAC does not scale well for a large number of vectors. This is because the time and space complexity for HAC is at least quadratic in the number of vectors (Manning et al., 2008).

We extracted more than three million vectors from the TAC source collection, a number well beyond our capacity for running the HAC algorithm. SONEX however implements the buckshot algorithm (Cutting et al., 1992). This algorithm reduces the time and space complexity of the HAC by clustering a sample of the three million vectors. By running a quadratic clustering algorithm over a sample of size $\sqrt{N}$,where N is the total number of vectors, the buckshot algorithm is able to find cluster centroids in linear time and space. It is well-accepted that the centroids of clusters produced from a representative sample are often as good as the centroids of the clusters produced from all vectors (Cutting et al., 1992).

Once we have the cluster centroids, we assign a cluster to each vector. We adopt the assign-to-nearest method, where the assigned cluster is the one that maximizes the similarity between the vector and the cluster centroid. We label each cluster with the token in the cluster centroid with the highest weight.

4.2 Mapping clusters to slots

We use the cluster label that SONEX provides to map a cluster to its appropriate slot (if any). To do so, we manually construct a list of valid labels for each slot by observing which clusters generated from the training data correspond to which slots. For example, a cluster with the label "husband" can be mapped to the `per:spouse` slot, so "husband" is added to the list of allowed labels for `per:spouse`. We were unable to find clusters for 5 slots, which forced us to submit NIL answers for them. The slots are: `per:cause_of_death`, `per:title`,

[4] http://mahout.apache.org/

per:charges, org:website and org:political_religious_affiliation.

5 YRES

Similar to SONEX, YRES looks within a certain context of a pair of entities, but considers words around the entities in the text as well as in between them. YRES applies an approach based on relative entropy as well as one that manually finds appropriate keywords for slots.

We hypothesize that if a word occurs more often in contexts for person/person pairs than for person/organization pairs, the word is likely to be indicative of a personal relationship between the entities. We refer to these pair classes (person/person, etc.) as *pair types*. To exploit this, YRES employs relative entropy $RE = p(x) \log \frac{p(x)}{q(x)}$, where for a word x, $p(x)$ is the frequency of x in contexts of a given pair type and $q(x)$ is the frequency of x in contexts of another pair type. Words with high relative entropy values are chosen with a manually-set threshold as keywords, which are manually mapped to slots for later use. We find that this method works well with person/person pair types for p and person/organization pair types for q, as well as vice versa, but other pair types are too noisy.

For these other pair types, YRES employs a simpler approach. Using the example training queries, entity pairs are generated using the various queries and the filled-in slot values. For each query-value pair, all sentences containing these pairs are gathered. Keywords are then manually chosen based on the terms appearing most frequently in the sentences mentioning the given pair and the keyword is mapped to the appropriate slot (based on the query and the value being examined). For example, the term "born" is quite frequent for the per:date_of_birth slot, and is therefore marked as a keyword mapping to that slot.

Finally, to extract relations from text, an entity pair is marked with slots based on the presence of relevant keywords. As defined above, an entity pair consists of all sentences mentioning a pair of entities; if a keyword (as found above) appears often (for some threshold defined manually based on the keyword) among these sentences, then the pair is marked as having the relationship (slot) corresponding to the keyword. When searching for the keyword, specific "context shapes" are defined for the various slots: for example, the per:date_of_birth slot looks for a keyword before and in between the two entities, while the org:employees slot looks in between and after; these windows are determined manually based on where the keywords tend to appear for the particular slots. Note that multiple keywords can be triggered, so a given pair can be marked as having multiple relationships.

6 Post-processing

To convert the relations extracted by SONEX and YRES to answers for slots, we apply a series of rules.

Because the relation extraction systems provide entities as they occur in the text, there must be some method of matching names found by the systems to those queried when there is not an exact match. For example, sometimes a name may include a title such as "Congressman" or "Senator", which would mean that exact matches would not work if given the name only as a query. We use a taboo list of titles, constructed from the Wikipedia "List of Titles" page[5], and ignore any words at the start of an entity name that appear on the list, so long as this leaves at least two words in the name. Similarly, we have a manually-constructed list of taboo endwords for organizations (including "corp.", "ltd.", etc.).

Both SONEX and YRES simply provide pairs of entities, which we use to construct a list of entities and slot values. While some slots imply a symmetric relation (per:spouse, for example), direction is distinctive for others (such as per:parents). Given a pair of entities in a given order, then, we first add the pairs to the entity-value list in that order; for instance, given "Barack Obama" and "Michelle Obama" with the per:spouse slot, we add "Michelle Obama" as a spouse of "Barack Obama". We then swap the order, which requires a different slot to be substituted in case direction is important. Of the slots considered, this means that per:children and per:parents

[5] https://secure.wikimedia.org/wikipedia/en/wiki/List_of_titles

System	Precision	Recall	F-measure
SONEX	0.3684	0.0518	0.0909
YRES	0.2602	0.0878	0.1313

Figure 1: TAC results for YRES and SONEX.

are swapped, as is the case with `org:parents` and `org:subsidiaries`. We must also verify that the two entities are of valid types so that we don't return people for the `org:parents` slot, etc.

Given a query, we can then look up the entity in our entity-value list, keeping in mind the taboo lists. We perform some simple checks on the location-valued slots such as `per:city_of_birth` and `per:origin`. Using the MONDIAL database[6], we ensure that the location value is a valid location and use MONDIAL to determine which slot (city, province, or country) should be filled. For the `per:origin` slot, we infer from cities or provinces to countries; if there are multiple cities or provinces with the given name, we choose the one with the highest population. We also perform checks for religion against a manually-constructed list of allowable religions or denominations.

Finally, once we have the slots filled for a given query, we need to adjust for single-valued slots as well as merge similar values together. For person-values, we group together all values having the same first name since the person slots that we fill are all familial. We also employ the taboo list for both persons and organizations as above. To pick a final value for a group, we choose the longest string to provide the most specific value possible (e.g. a full name instead of a first name only), reverting to system scores in case of a tie.

7 Results

We submitted two runs, one for YRES and another one for SONEX. Figure 1 shows the results for these submissions. We analyze the results in the following sections.

7.1 YRES

Ignoring slot types which have less than 10 entries in the standard result, the following are some specific statistics. The slot we miss the most is *org:alternate_names*, for which we extracted no answer. Slots with the maximum number of correct answers are *per:title* and *org:top_members/employees*. The slot with the maximum number of wrong answers is *per:member_of*, with 8 correct and 57 wrong. After much scrutiny we found there are two major types of error causes. One is that the relation word is not related to the entity pairs. For example, in "PER told leaders of ORG...", "leader" is not related to the "PER–ORG" pair. The other is named entity recognition, the entity names are mismatched or incomplete. For instance, "Democratic Senator John Kerry", the ORG "Democratic" is incomplete, which should be "Democratic party".

[6]`http://www.dbis.informatik.uni-goettingen.de/Mondial/`

Slot	Precision
ORG:Number of Employees	0.95
PER:Employee of	0.92
PER:Member of	0.92
ORG:Headquarters	0.90
PER:Place of birth	0.88
...	
PER:Children	0.66
ORG:Subsidiaries	0.65
ORG:Alternate Names	0.56
ORG:Parents	0.55
ORG:Shareholders	0.50
Average	0.79

Figure 2: The slots in descendant order of precision as measured in the preliminary experiment. For brevity, only the top and bottom five are shown.

7.2 SONEX

Preliminary results. We conducted a preliminary experiment to estimate the precision of the clusters produced by SONEX. We randomly selected 50 entity pairs from clusters mapped to slots. We asked four students to verify whether the entity pairs of each slot were supported by any document in source collection. Entity pairs with a supporting document were deemed as correct. Figure 2 shows the precision for ten slots, being five with highest score and five with the lowest. The average precision for all slots is 0.79.

TAC results. Figure 1 shows the scores for the SONEX submission. SONEX was able to correctly answer 49 fillers (out of 945 correct answers). One could expect a low recall score from SONEX, given its limitation of extracting exactly one relation for each entity pair (as oppose to all relations between a pair of entities).

While low recall was expected, we were surprised by the low precision score in our TAC submission (in light of our preliminary experiment). In order to understand the discrepancy between the results from TAC and our experiment, we looked carefully at the answers not judged as correct. Out of 84 non-correct answers, 62 are incorrect, 15 are redundant and 7 are inexact. The majority of incorrect answers are from the slots `per:employee_of` (21) and `per:top_members_employees` (13). Analyzing the slot fillers for these two slots, we have found that 38% of the incorrect answer in these slots are due to mistakes when handling entities. For example, our coreference resolver incorrectly determined that "U.N." (United Nations) and "US Navy" refers to the same entity. Observe that the initials of "<u>U</u>S <u>N</u>avy" match "U.N.". This coreference decision resulted in a wrong extraction from the text portion "U.N. nuclear watchdog chief Mohamed ElBaradei". In particular, SONEX answered "US Navy" (instead of "U.N.") for the slot `per:employee_of` of the entity "Mohamed ElBaradei". We observed that mistakes that arise from mishandling entities are not equally distributed through the entities. That is, some entities are more likely to be mishandled. The random samples used in our preliminary experiment is unlikely to show this distribution. This is because we selected individual entity pairs as oppose to all slot fillers for a target entity.

8 Future work & conclusion

We have presented two approaches to the slot-filling task. YRES identifies slot fillers by leveraging keywords indicative of each slot. SONEX relies on open relation extraction, where the relations to be extracted are assumed to be unknown *a priori*. We manually mapped each slot (e.g., `per:spouse`) to corresponding relations extracted by SONEX (e.g., wife, husband). SONEX (0.09) and YRES (0.13) achieved similar F-measure scores. SONEX presented a higher precision at expense of a lower recall.

To increase the recall of both system systems, we are considering expanding the context of an entity pair to include the text windows appearing before and after the pair in all sentences. In addition, we are investigating the impact of adding prepositions into the context of an entity pair, especially when nouns and verbs are not present. In order to improve precision, we plan to train a classifier to identify a whether two entities are related in a sentence.

The plans for future work also include reducing the number of incorrect extractions due to entity mishandling by combining different named entity recognition and coreference resolution systems. We will also investigate how we can improve the entity type classification by linking the entities found in the corpus to those in a knowledge base.

We plan to replace SONEX's feature extraction step, which consider individual tokens, by a more sophisticated one. In particular, we are investigating techniques to extract relational phrases (e.g., "was acquired by") as oppose to individual tokens only.

Finally, both YRES and SONEX are unable to recognize the *directionality* of some relations. By directionality, we mean the correct order of the entities in a relation. For example, we expect that the first entity of relation "children" to be a parent, while the second entity should be a child. We plan to study methods to discover the correct directionally of a relation even when the entities are shown in reverse order (e.g., "C, children of P").

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: extracting relations from large plain-text collections. In *Proceedings of the ACM Conference on Digital libraries*, pages 85–94. ACM.

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 28–36, Columbus, Ohio, June. Association for Computational Linguistics.

Kalina Bontcheva, Marin Dimitrov, Diana Maynard, Valentin Tablan, and Hamish Cunningham. 2002. Shallow methods for named entity coreference resolution. In *Chaines de references et resolveurs d'anaphores, workshop TALN*, June.

Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *Proceedings of the World Wide Web and Databases (WebDB) International Workshop*, pages 172–183.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In Raymond J. Mooney, editor, *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731. Association for Computational Linguistics.

Aron Culotta and Jeffrey S. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 423–429. Association for Computational Linguistics.

Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. 1992. Scatter/gather: a cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '92, pages 318–329, New York, NY, USA. ACM.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the International conference on World Wide Web*, pages 100–110, New York, NY, USA. ACM.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, pages 363–370, Ann Arbor, MI, USA, June. Association for Computational Linguistics.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 427–434. Association for Computational Linguistics.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, page 415. Association for Computational Linguistics.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, 1 edition, July.

Filipe Mesquita, Yuval Merhav, and Denilson Barbosa. 2010. Extracting information networks from the blogosphere: State-of-the-art and challenges. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM), Data Challenge Workshop*, Washington D.C., 05/2010.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106.

Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. 2009. Statsnowball: a statistical approach to extracting entity relationships. In *Proceedings of the International Conference on World Wide Web*, pages 101–110. ACM.