Semi-Supervised Bootstrapping of Relationship Extractors with Distributional Semantics

David S. Batista **Bruno Martins** **Mário J. Silva**
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
{david.batista,bruno.g.martins,mario.gaspar.silva}@ist.utl.pt

Abstract

Semi-supervised bootstrapping techniques for relationship extraction from text iteratively expand a set of initial seed relationships while limiting the semantic drift. We research bootstrapping for relationship extraction using word embeddings to find similar relationships. Experimental results show that relying on word embeddings achieves a better performance on the task of extracting four types of relationships from a collection of newswire documents when compared with a baseline using TF-IDF to find similar relationships.

1 Introduction

Relationship Extraction (RE) transforms unstructured text into relational triples, each representing a relationship between two named-entities. A bootstrapping system for RE starts with a collection of documents and a few seed instances. The system scans the document collection, collecting occurrence contexts for the seed instances. Then, based on these contexts, the system generates extraction patterns. The documents are scanned again using the patterns to match new relationship instances. These newly extracted instances are then added to the seed set, and the process is repeated until a certain stop criteria is met.

The objective of bootstrapping is thus to expand the seed set with new relationship instances, while limiting the semantic drift, i.e. the progressive deviation of the semantics for the extracted relationships from the semantics of the seed relationships.

State-of-the-art approaches rely on word vector representations with TF-IDF weights (Salton and Buckley, 1988). However expanding the seed set by relying on TF-IDF representations to find similar instances has limitations, since the similarity between any two relationship instance vectors of TF-IDF weights is only positive when the instances share at least one term. For instance, the phrases *was founded by* and *is the co-founder of* do not have any common words, but they have the same semantics. Stemming techniques can aid in these cases, but only for variations of the same root word (Porter, 1980).

We propose to address this challenge with an approach based on word embeddings (Mikolov et al., 2013a). By relying on word embeddings, the similarity of two phrases can be captured even if no common words exist. The word embeddings for *co-founder* and *founded* should be similar, since these words tend to occur in the same contexts. Word embeddings can nonetheless also introduce semantic drift. When using word embeddings, phrases like *studied history at* can, for instance, have a high similarity with phrases like *history professor at*. In our approach, we control the semantic drift by ranking the extracted relationship instances, and by scoring the generated extraction patterns.

We implemented these ideas in BREDS, a bootstrapping system for RE based on word embeddings. BREDS was evaluated with a collection of 1.2 million sentences from news articles. The experimental results show that our method outperforms a baseline bootstrapping system based on the ideas of Agichtein and Gravano (2000) which relies on TF-IDF representations.

2 Bootstrapping Relationship Extractors

Brin (1999) developed DIPRE, the first system to apply bootstrapping for RE, which represents the occurrences of seeds as three contexts of strings: words before the first entity (BEF), words between the two entities (BET), and words after the second entity (AFT). DIPRE generates extraction patterns by grouping contexts based on string matching, and controls semantic drift by limiting the number of instances a pattern can extract.

Agichtein and Gravano (2000) developed Snowball, which is inspired on DIPRE's method of collecting three contexts for each occurrence, but computing a TF-IDF representation for each context. The seed contexts are clustered with a single-pass algorithm based on the cosine similarity between contexts using the three vector representations:

$$\begin{aligned}\text{Sim}(S_n, S_j) = {} & \alpha \cdot \cos(BEF_i, BEF_j) \\ & + \beta \cdot \cos(BET_i, BET_j) \\ & + \gamma \cdot \cos(AFT_i, AFT_j)\end{aligned} \tag{1}$$

In the formula, the parameters α, β and γ weight each vector. An extraction pattern is represented by the centroid of the vectors that form a cluster. The patterns are used to scan the text again, and for each segment of text where any pair of entities with the same semantic types as the seeds co-occur, three vectors are generated. If the similarity from the context vectors towards an extraction pattern is greater than a threshold τ_{sim}, the instance is extracted.

Snowball scores the patterns and ranks the extracted instances to control the semantic drift. A pattern is scored based on the instances that it extracted, which can be included in three sets: P, N, and U. If an extracted instance contains an entity e_1, which is part of a seed, and if the associated entity e_2 in the instance is the same as in in the seed, then the extraction is considered positive (included in set P). If the relationship contradicts a relationship in the seed set (i.e., e_2 does not match), then the extraction is considered negative (included in a set N). If the relationship is not part of the seed set, the extraction is considered unknown (included in a set U). A score is assigned to each pattern p according to:

$$\text{Conf}_\rho(p) = \frac{|P|}{|P| + W_{ngt} \cdot |N| + W_{unk} \cdot |U|} \tag{2}$$

W_{ngt} and W_{unk} are weights associated to the negative and unknown extractions, respectively. The confidence of an instance is calculated based on the similarity scores towards the patterns that extracted it, weighted by the pattern's confidence:

$$\text{Conf}_\iota(i) = 1 - \prod_{j=0}^{|\xi|}(1 - \text{Conf}_\rho(\xi_j) \times \text{Sim}(C_i, \xi_j)) \tag{3}$$

where, ξ is the set of patterns that extracted i, and C_i is the textual context where i occurred. Instances with a confidence above a threshold τ_t are used as seeds in the next iteration.

3 Bootstrapping Relationship Extractors with Word Embeddings

BREDS follows the architecture of Snowball, having the same processing phases: find seed matches, generating extraction patterns, finding relationship instances, and detecting semantic drift. It differs, however, in that it attempts to find similar relationships using word embeddings, instead of relying on TF-IDF representations.

3.1 Find Seed Matches

BREDS scans the document collection and, if both entities of a seed instance co-occur in a text segment within a sentence, then that segment is considered and BREDS extracts the three textual contexts as in Snowball: BEF, BET, and AFT.

In the BET context, BREDS tries to identify a relational pattern based on a shallow heuristic originally proposed in ReVerb (Fader et al., 2011). The pattern limits a relation context to a verb (e.g., *invented*), a verb followed by a preposition (e.g., *located in*), or a verb followed by nouns, adjectives, or adverbs ending in a preposition (e.g., *has atomic weight of*). These patterns will nonetheless only consider verb mediated relationships. If no verbs exist between two entities, BREDS extracts all the words between the two entities, to build the representations for the BET context.

Each context is transformed into a single vector by a simple compositional function that starts by removing stop-words and adjectives and then sums the word embedding vectors of each individual word. Representing small phrases by summing each individual word's embedding results in good representations for the semantics in small phrases (Mikolov et al., 2013b).

A relationship instance i is represented by three embedding vectors: V_{BEF}, V_{BET}, and V_{AFT}. Considering the sentence:

The tech company Soundcloud *is based in* Berlin, *capital of Germany.*

BREDS generates the relationship instance with:

$$V_{BEF} = \text{E}(\text{“tech”}) + \text{E}(\text{“company”})$$
$$V_{BET} = \text{E}(\text{“is”}) + \text{E}(\text{“based”})$$
$$V_{AFT} = \text{E}(\text{“capital”})$$

where, E(x) is the word embedding for word x.

BREDS also tries to identify the passive voice using part-of-speech (PoS) tags, which can help to detect the correct order of the entities in a relational triple. BREDS identifies the presence of the passive voice by considering any form of the verb *to be*, followed by a verb in the past tense or the past participle, and ending in the word *by*.

For instance, the seed `<Google, owns, DoubleClick>` states that the organisation `Google` owns the organisation `DoubleClick`. Using this seed, if BREDS detects a pattern like *agreed to be acquired by* it will swap the order of the entities when producing a relational triple, outputting the triple <ORG$_2$, *owns*, ORG$_1$>, instead of the triple <ORG$_1$, *owns*, ORG$_2$>.

3.2 Extraction Patterns Generation

As Snowball, BREDS generates extraction patterns by applying a single-pass clustering algorithm to the relationship instances gathered in the previous step. Each resulting cluster contains a set of relationship instances, represented by their three context vectors.

Algorithm 1 describes the clustering approach taken by BREDS, which takes as input a list of relationship instances and assigns the first instance to a new empty cluster. Next, it iterates through the list of instances, computing the similarity between an instance i_n and every cluster Cl_j. The instance i_n is assigned to the first cluster whose similarity is higher or equal to a threshold τ_{sim}. If all the clusters have a similarity lower than a threshold τ_{sim}, a new cluster C_m is created, containing the instance i_n.

The similarity function Sim(i_n, Cl_j), between an instance i_n and a cluster Cl_j, returns the maximum of the similarities between an instance i_n and any of the instances in a cluster Cl_j, if the majority of the similarity scores is higher than a threshold τ_{sim}. A value of zero is returned otherwise. The similarity between two instances is computed according to Formula (1). As a result, clustering in Algorithm 1 differs from the original Snowball method, which instead computes similarities towards cluster centroids.

3.3 Find Relationship Instances

After the generation of extraction patterns, BREDS finds relationship instances with Algorithm 2. It scans the documents once again, collecting all segments of text containing entity pairs whose semantic types are the same as those in the seed instances. For each segment, an instance i is generated as described in Section 3.1, and the similarity towards all previously generated extraction patterns (i.e., clusters) is computed. If the similarity between i and a pattern Cl_j is equal or above τ_{sim}, then i is considered a candidate instance, and the confidence score of the pattern is updated, according to Formula (2). The pattern which has the highest similarity ($pattern_{best}$) is associated with i, along with the corresponding similarity score (sim_{best}). This information is kept in a history of $Candidates$. Note that the histories of $Candidates$ and $Patterns$ are kept through all the bootstrap iterations, and new patterns or instances can be added, or the scores of existing patterns or instances can change.

Algorithm 1: Single-Pass Clustering.

```
Input: $Instances = \{i_1, i_2, i_3, ..., i_n\}$
Output: $Patterns = \{\}$
$Cl_1 = \{i_1\}$
$Patterns = \{Cl_1\}$
for $i_n \in Instances$ do
    for $Cl_j \in Patterns$ do
        if $Sim(i_n, Cl_j) >= \tau_{sim}$ then
            $Cl_j = Cl_j \cup \{i_n\}$
        else
            $Cl_m = \{i_n\}$
            $Patterns = Patterns \cup \{Cl_m\}$
```

Algorithm 2: Find Relationship Instances.

```
Input: $Sentences = \{s_1, s_2, s_3, ..., s_n\}$
Input: $Patterns = \{Cl_1, Cl_2, ..., Cl_n\}$
Output: $Candidates$
for $s_i \in Sentences$ do
    $i = create\_instance(s_i)$
    $sim_{best} = 0$
    $p_{best} = None$
    for $Cl_i \in Patterns$ do
        $sim = \text{Sim}(i, Cl_i)$
        if $sim >= \tau_{sim}$ then
            $\text{Conf}_\rho(C_i)$
            if $sim >= sim_{best}$ then
                $sim_{best} = sim$
                $P_{best} = Cl_i$
    $Candidates[i].patterns[p_{best}] = sim_{best}$
```

3.4 Semantic Drift Detection

As Snowball, BREDS ranks the candidate instances at the end of each iteration, based on the scores computed with Formula (3). Instances with a score equal or above the threshold τ_t are added to the seed set, for use in the next iteration of the bootstrapping algorithm.

4 Evaluation

In our evaluation we used a set of 5.5 million news articles from AFP and APW (Parker et al., 2011).

Our pre-processing pipeline is based on the models provided by the NLTK toolkit (Bird et al., 2009): sentence segmentation[1], tokenisation[2], PoS-tagging[3] and named-entity recognition (NER). The NER module in NLTK is a wrapper over the Stanford NER toolkit (Finkel et al., 2005).

We performed weak entity-linking by matching entity names in sentences with FreebaseEasy (Bast et al., 2014). FreebaseEasy is a processed version of Freebase (Bollacker et al., 2008), which contains a unique meaningful name for every entity, together with canonical binary relations. For our experiments, we selected only the sentences containing at least two entities linked to FreebaseEasy, which corresponded to 1.2 million sentences.

With the full articles set, we computed word embeddings with the skip-gram model[4] using the *word2vec*[5] implementation from Mikolov et. al. (2013a). The TF-IDF representations used by Snowball were calculated over the same articles set. We adopted a previously proposed framework for the evaluation of large-scale RE systems by Bronzi et al. (2012), to estimate precision and recall, using FreebaseEasy as the knowledge base.

We considered entity pairs no further away than 6 tokens, and a window of 2 tokens for the BEF and AFT contexts, ignoring the remaining of the sentence. We discarded the clusters with only one relationship instances, and ran a maximum of 4 bootstrapping iterations. The W_{unk} and W_{ngt} parameters were set to 0.1 and 2, respectively, based on the results reported by Yu et al. (2003).

We compared BREDS against Snowball in four relationship types, shown in Table 1. For each relationship type we considered several bootstrapping configurations by combining different values for the τ_{sim} and τ_t thresholds, all within the interval [0.5,1.0].

Relationship	**Seeds**
acquired	{Adidas, Reebok} {Google, DoubleClick}
founder-of	{CNN, Ted Turner} {Amazon, Jeff Bezos}
headquartered	{Nokia, Espoo} {Pfizer, New York}
affiliation	{Google, Marissa Mayer} {Xerox, Ursula Burns}

Table 1: Relationship types and used seeds.

We bootstrapped each relationship with two context weighting configurations in Formula (1):

- Conf_1: $\alpha = 0.0, \beta = 1.0, \gamma = 0.0$
- Conf_2: $\alpha = 0.2, \beta = 0.6, \gamma = 0.2$

where Conf_1 only considers the BET context and Conf_2 uses the three contexts, while giving more importance to the BET context.

Table 2 shows, for each relationship type, the best F_1 score and the corresponding precision and recall, for all combinations of τ_{sim} and τ_t values, and considering only extracted relationship instances with confidence scores equal or above 0.5. Table 2a shows the results for the BREDS system, while Table 2b shows the results for Snowball (ReVerb), a modified Snowball in which a relational pattern based on ReVerb is used to select the words for the BET context. Finally, Table 2c shows the results for Snowball, implemented as described in the original paper.

Overall, BREDS achieves better F_1 scores than both versions of Snowball. The F_1 score of BREDS is higher, mainly as a consequence of much higher recall scores, which we believe to be due to the relaxed semantic matching caused by using the word embeddings. For some relationship types, the recall more than doubles when using word embeddings instead of TF-IDF. For the *acquired* relationship, when considering Conf_1, the precision of BREDS drops compared with the other versions of Snowball, but without affecting the F_1 score, since the higher recall compensates for the small loss in precision.

Regarding the context weighting configurations, Conf_2 produces a lower recall when compared to Conf_1. This might be caused by the

[1]nltk.tokenize.punkt.PunktSentenceTokenizer

[2]nltk.tokenize.treebank.TreebankWordTokenizer

[3]taggers/maxent_treebank_pos_tagger/english.pickle

[4]skip length of 5 tokens and vectors of 200 dimensions

[5]`https://code.google.com/p/word2vec/`

sparsity of both BEF and AFT, which contain many different words that do not contribute to capture the relationship between the two entities. Although, sometimes, the phrase or word that indicates a relationship occurs on the BEF or AFT contexts, it is more often the case that these phrases or words occur in the BET context.

The performance results of Snowball (Classic) and Snowball (ReVerb) suggest that selecting words based on a relational pattern to represent the BET context, instead of using all the words, works better for TF-IDF representations.

The results also show that word embeddings can generate more extraction patterns. For instance, for the *founder-of* relationship, BREDS learns patterns based on words such as *founder, co-founder, co-founders* or *founded*, while Snowball only learns patterns that have the word *founder*, like *CEO and founder* or *founder and chairman*.

The implementations of BREDS and Snowball, as described in this paper, are available on-line[6].

5 Conclusions and Future Work

This paper reports on a novel bootstrapping system for relation extraction based on word embeddings. In our experiments, bootstrapped RE achieved better results when using word embeddings to find similar relationships than with similarities between TF-IDF weighted vectors.

We have identified two main sources of errors: NER problems and incorrect relational patterns extraction due to the use of a shallow heuristic that only captures local relationships.

In future work, more robust entity-linking approaches, as proposed by Hoffart et al. (2011), could be included in our pre-processing pipeline. This could alleviate NER errors and enable experimentation with other relationship types.

Gabbard et al. (2011) have shown that co-reference resolution can increase bootstrapping RE performance, and the method of Durrett and Klein (2014) could also be included in our pre-processing pipeline.

Finally, we could explore richer compositional functions, combining word embeddings with syntactic dependencies (SD) (Yu et al., 2014). The shortest path between two entities in an SD tree supports the extraction of local and long-distance relationships (Bunescu and Mooney, 2005).

[6] `https://github.com/davidsbatista/BREDS`

BREDS			
Relationship	**Precision**	**Recall**	**F_1**
	Conf$_1$		
acquired	0.73	**0.77**	**0.75**
founder-of	**0.98**	**0.86**	**0.91**
headquartered	0.63	**0.69**	**0.66**
affiliation	**0.85**	**0.91**	**0.88**
	Conf$_2$		
acquired	**1.00**	0.15	0.26
founder-of	0.97	0.79	0.87
headquartered	**0.64**	0.61	0.62
affiliation	0.84	0.60	0.70

(a) Precision, Recall and F_1 results obtained with different configurations of BREDS.

Snowball (ReVerb)			
Relationship	**Precision**	**Recall**	**F_1**
	Conf$_1$		
acquired	0.83	0.61	0.70
founder-of	0.96	0.77	0.86
headquartered	0.48	0.63	0.55
affiliation	0.52	0.29	0.37
	Conf$_2$		
acquired	0.73	0.22	0.34
founder-of	0.97	0.75	0.85
headquartered	0.55	0.42	0.47
affiliation	0.36	0.05	0.08

(b) Precision, Recall and F_1 results obtained with different configurations of Snowball (ReVerb).

Snowball (Classic)			
Relationship	**Precision**	**Recall**	**F_1**
	Conf$_1$		
acquired	0.87	0.54	0.67
founder-of	0.97	0.76	0.85
headquartered	0.52	0.61	0.57
affiliation	0.49	0.29	0.36
	Conf$_2$		
acquired	0.77	0.54	0.63
founder-of	0.98	0.73	0.84
headquartered	0.53	0.54	0.54
affiliation	0.42	0.08	0.13

(c) Precision, Recall and F_1 results obtained with different configurations of Snowball (Classic).

Table 2: Precision, Recall and F_1 scores over the four relationships for the three different systems.

Acknowledgements

This work was supported by Fundação para a Ciência e a Tecnologia, under grants SFRH/BD/70478/2010, UID/CEC/50021/2013 and EXCL/EEI- ESS/0257/2012 (DataStorm).

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting Relations from Large Plain-Text Collections. In *Proceedings of the ACM Conference on Digital libraries*, DL'00.

Hannah Bast, Florian Bäurle, Björn Buchhold, and Elmar Haußmann. 2014. Easy Access to the Freebase Dataset. In *Companion Publication of the 23rd International Conference on World Wide Web*, WWW Companion '14.

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

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08.

Sergey Brin. 1999. Extracting Patterns and Relations from the World Wide Web. In *Selected Papers from the International Workshop on The World Wide Web and Databases*, WebDB'98.

Mirko Bronzi, Zhaochen Guo, Filipe Mesquita, Denilson Barbosa, and Paolo Merialdo. 2012. Automatic Evaluation of Relation Extraction Systems on Large-scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*, HLT/EMNLP'05.

Greg Durrett and Dan Klein. 2014. A Joint Model for Entity Analysis: Coreference, Typing, and Linking. *Transactions of the Association for Computational Linguistics*, 2.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, ACL'05.

Ryan Gabbard, Marjorie Freedman, and Ralph M. Weischedel. 2011. Coreference for Learning to Extract Relations: Yes Virginia, Coreference Matters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, ACL'11.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the Workshop at the International Conference on Learning Representations*, ICLR'13.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the Conference on Neural Information Processing Systems*, NIPS'13.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition LDC2011T07.

M.F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3).

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5).

Hong Yu and Eugene Agichtein. 2003. Extracting Synonymous Gene and Protein Terms from Biological Literature. *Bioinformatics*, 19(suppl 1).

Mo Yu, Matthew R. Gormley, and Mark Dredze. 2014. Factor-based Compositional Embedding Models. In *Proceedings of the Conference on Neural Information Processing Systems*, NIPS'14.