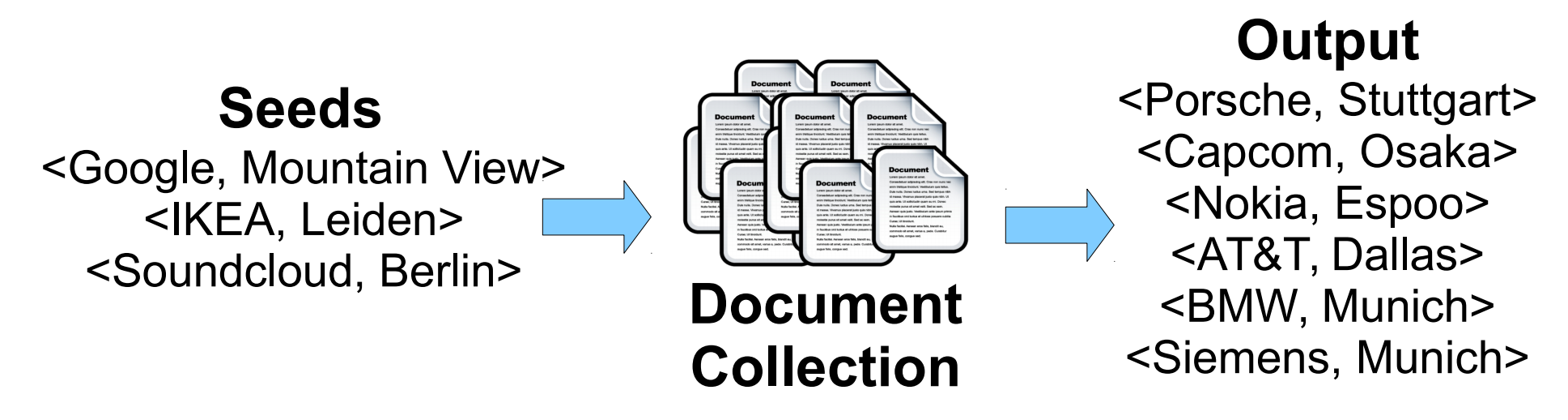
Semi-Supervised Bootstrapping of Relationship Extractors with Distributional Semantics

David S. Batista Bruno Martins Mário J. Silva
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa

From seeds extract new relationship instances.



Based on generating a context representation for each pair of entities, in order to find similar contexts.

Snowball

TF-IDF vector weighting

X = "main headquarters in"	1.3	2.6	0	1.7	0	0.8
Y = "is based in"	0	0	0	0	3.3	0
Z = "is headquartered in"	0	0	2.2	0	0	0

Although the phrases have the same semantics:
cos(X,Y) = cos(X,Z) = cos(Y,Z) = 0

IDEA:

- Identify the word(s) that mediate a relationship between two named entities.
- Use embeddings of these words to represent the relationship context.
- Embeddings can capture semantic similarity

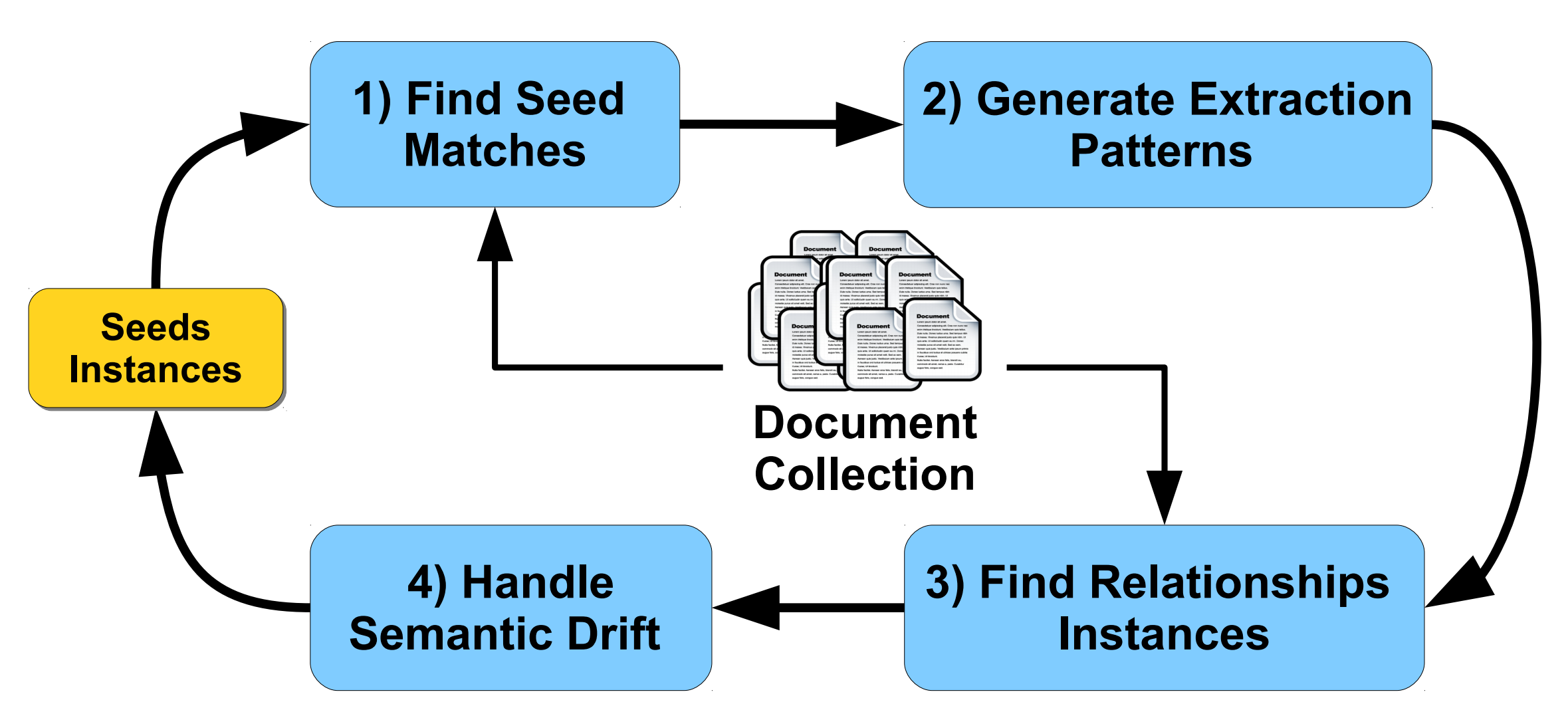
BREDS

Word embeddings

"headquarters"	0.18	0.22	0.82	0.65	0.33	0.23
"based"	0.16	0.76	0.81	0.63	0.31	0.33
"headquartered"	0.22	0.81	0.81	0.64	0.36	0.33

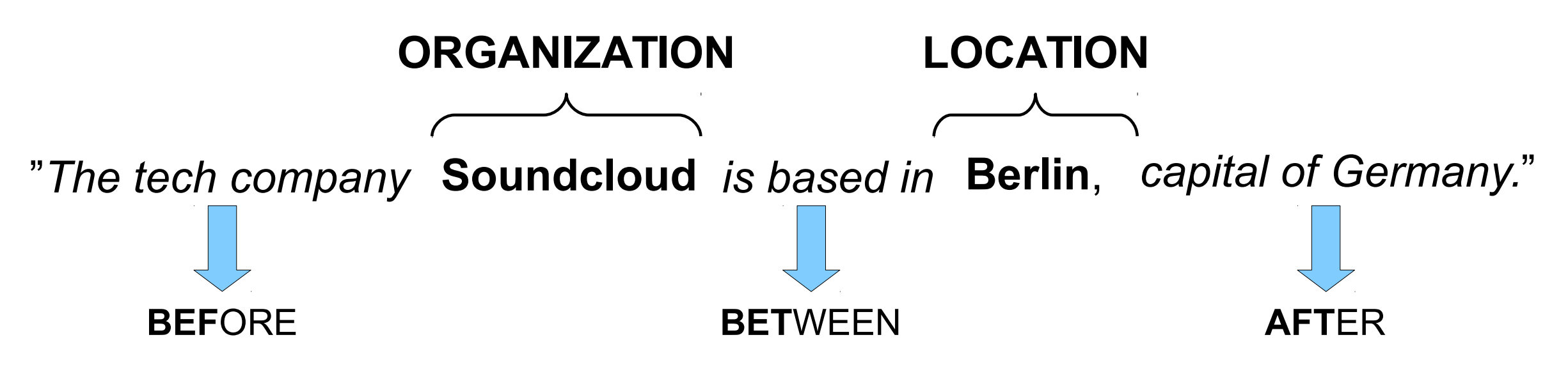
- Embeddings for *headquartered*, *based*, or *headquartered* should be similar, since these words tend to occur in the same contexts.

General procedure



1) Find Seed Matches

1) Start with a few seed instances of a relationship type (e.g., *headquartered*), find text segments where they co-occur, and extract 3 contexts.


ORGANIZATION
LOCATION
"The tech company Soundcloud is based in Berlin, capital of Germany."
BEFORE
BETWEEN
AFTER

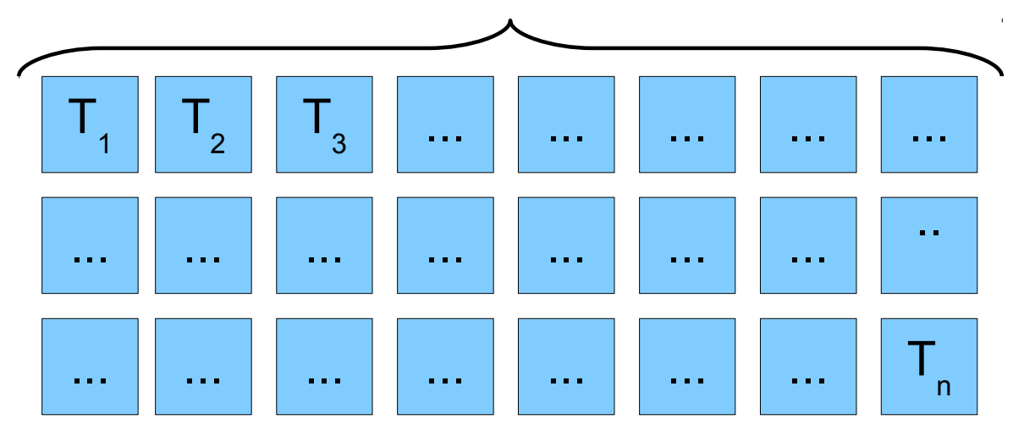
2) Look for ReVerb relational patterns in the BETWEEN context, based on PoS-tags.

3) Transform each context into an embedding vector with a simple compositional function that removes stop-words and adjectives, and then sums the embeddings of each word.

$$T_n \left\{ \begin{array}{l} Vector_{BEFORE} = E(''tech'') + E(''company'') \\ Vector_{BETWEEN} = E(''is'') + E(''based'') \\ Vector_{AFTER} = E(''capital'') \end{array} \right.$$

2) Generate Extraction Patterns

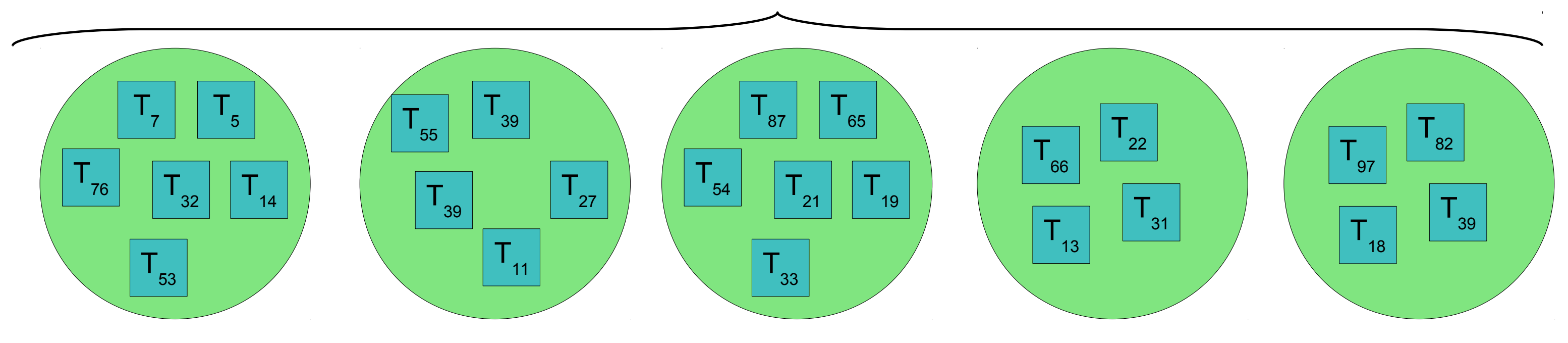
Collected seed instances



Single-Pass Clustering considering a threshold τ_{sim}

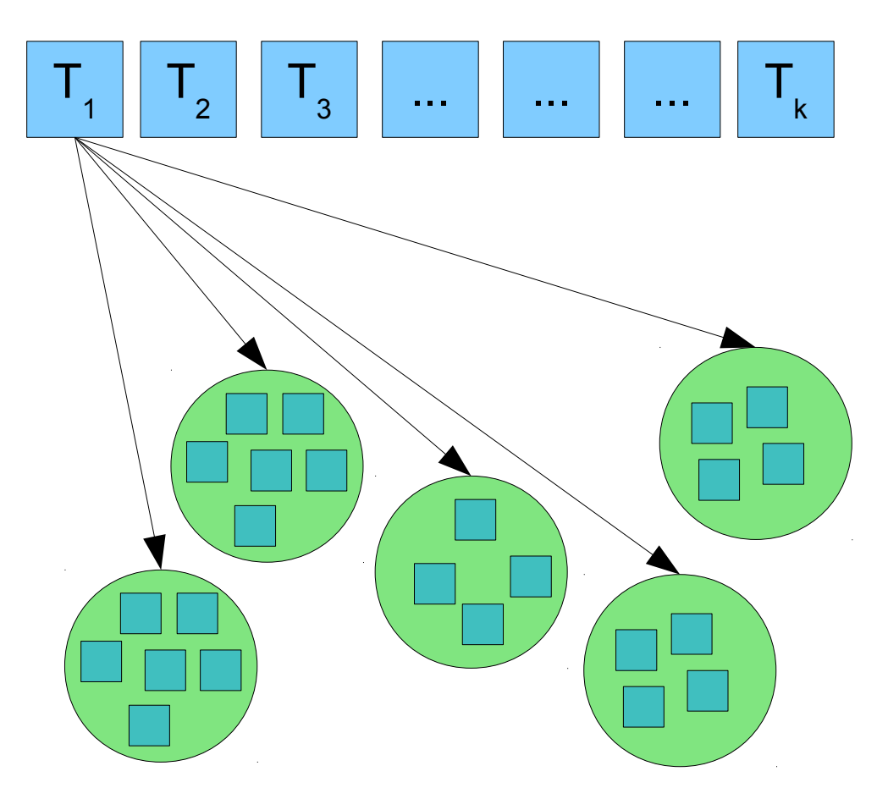
$$\mathrm{Sim}(T_i, T_j) = \alpha \cos(BEF_i, BEF_j) + \beta \cos(BET_i, BET_j) + \gamma \cos(AFT_i, AFT_j)$$

Generated Extraction Patterns (Clusters of instances)



3) Find Relationship Instances

Extract segments of text with the seed's semantic types (e.g., <ORG, LOC>) and generate the embeddings context (i.e., BEF, BET, AFT).



Compute the similarity of each instance towards every extraction pattern, associate best pattern and corresponding similarity score with that instance.

Rank patterns whose similarity with an instance is higher than threshold τ_{sim}

$$Conf(P) = \log_2(Positive) \times \left(\frac{Positive}{Positive + Uknown \times W_{unk} + Negative \times W_{ngt}} \right)$$

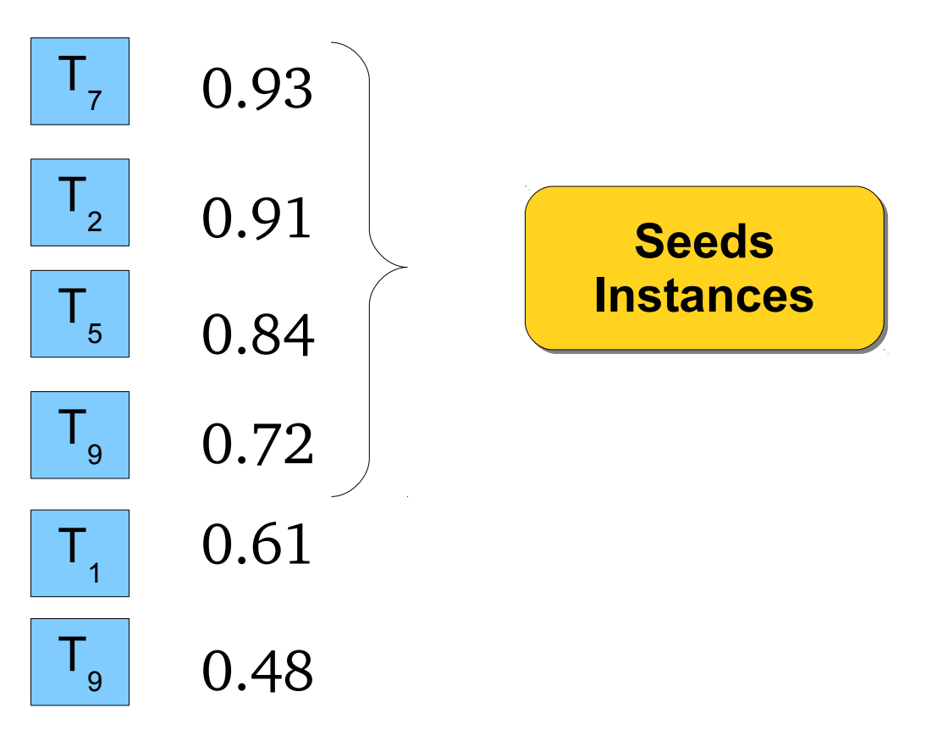
4) Handle Semantic Drift

Rank the extracted instances according to a confidence score, based on the patterns and similarity scores.

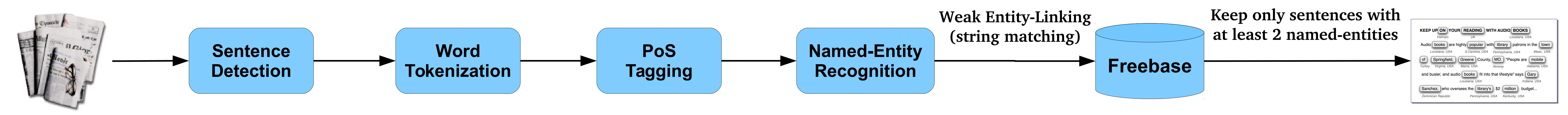
$$Conf(T) = 1 - \prod_{i=0}^{|P|} \left(1 - \left(Conf(P_i) \times Match(T, P_i)\right)\right)$$

Every instance whose confidence is above a threshold is added to the seed set and used in the next bootstrapping iteration.

$Conf(T) > \tau_t$



Experiments and Results


Sentence Detection
Word Tokenization
PoS Tagging
Named-Entity Recognition
Weak Entity-Linking (string matching)
Freebase
Keep only sentences with at least 2 named-entities

5.5 million news articles

Input corpus circa 1.2 million sentences

- With the 5.5 million articles we generated word embeddings with the skip-gram model (5 skip tokens, 200 dimensions vectors) and the TF-IDF weights.
- For each relationship we considered several runs combining different values [0.5,1.0] for the thresholds τ_{sim}, τ_t

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>

Two similarity weighing configurations

Weighting$_1$	Weighting$_2$
$\alpha = 0.0$	$\alpha = 0.2$
$\beta = 1.0$	$\beta = 0.6$
$\gamma = 0.0$	$\gamma = 0.2$

Results

BREDS				Snowball (ReVerb)				Snowball (Classic)			
Relationship	Precision	Recall	F$_1$	Relationship	Precision	Recall	F$_1$	Relationship	Precision	Recall	F$_1$
	Weighting$_1$				Weighting$_1$				Weighting$_1$		
acquired	0.73	**0.77**	**0.75**	acquired	0.83	0.61	0.70	acquired	0.87	0.54	0.67
founder-of	**0.98**	**0.86**	**0.91**	founder-of	0.96	0.77	0.86	founder-of	0.97	0.76	0.85
headquartered	0.63	**0.69**	**0.66**	headquartered	0.48	0.63	0.55	headquartered	0.52	0.61	0.57
affiliation	**0.85**	**0.91**	**0.88**	affiliation	0.52	0.29	0.37	affiliation	0.49	0.29	0.36
	Weighting$_2$				Weighting$_2$				Weighting$_2$		
acquired	**1.00**	0.15	0.26	acquired	0.73	0.22	0.34	acquired	0.77	0.54	0.63
founder-of	0.97	0.79	0.87	founder-of	0.97	0.75	0.85	founder-of	0.98	0.73	0.84
headquartered	**0.64**	0.61	0.62	headquartered	0.55	0.42	0.47	headquartered	0.53	0.54	0.54
affiliation	0.84	0.60	0.70	affiliation	0.36	0.05	0.08	affiliation	0.42	0.08	0.13

Conclusions

- BREDS achieves better F$_1$ scores mainly as a consequence of much higher recall, due to the relaxed semantic matching based on embeddings.
- Weighting$_2$ produces a lower recall due to the fact that both BEF and AFT contain many different words that do not contribute to capturing relationships between pairs of entities.
- Selecting words based on ReVerb relational patterns to represent the BET context, instead of using all words, works better for TF-IDF representations.

Future Work:
- Use a more robust entity-linking approach to alleviate NER errors.
- Explore richer compositional functions, combining word embeddings with syntactic dependencies.

Source code available: https://github.com/davidsbatista/BREDS

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