# JEDI: Joint Entity and Relation Detection using Type Inference

**Johannes Kirschnick[1], Holmer Hemsen[1], Volker Markl[1,2]**
[1]DFKI Project Office Berlin, Alt-Moabit 91c, Berlin
firstname.lastname@dfki.de
[2]Technische Univeristät Berlin
Database Systems and Information Management Group
Einsteinufer 17, 10587 Berlin, Germany
firstname.lastname@tu-berlin.de

## Abstract

FREEBASE contains entities and relation information but is highly incomplete. Relevant information is ubiquitous in web text, but extraction deems challenging. We present JEDI, an automated system to jointly extract typed named entities and FREEBASE relations using dependency pattern from text. An innovative method for constraint solving on entity types of multiple relations is used to disambiguate pattern. The high precision in the evaluation supports our claim that we can detect entities and relations together, alleviating the need to train a custom classifier for an entity type[1].

## 1 Introduction

Finding, tagging and extracting relations in web text is one of the more challenging tasks in Information Extraction (IE). It consists of correctly labeling entities as instances of a particular type (such as Person, Organization or Location) and detecting relations between them, such as *worksIn*, *bornIn* or even more fine grained ones such as *receiveDegree*. These relations are stored for further analysis in knowledge bases, but often existing ones are highly incomplete. Min et al. (2013) report that in the collaborative-edited FREEBASE[2] knowledge base $93.8\%$ of the person entities are missing a *place of birth* entry. To close the cap automated methods are needed that can detect these relations, by analyzing the abundance of web text.

A typical process to detect relations uses the linking words (pattern) between two entities to label a relation, but this poses the challenge of dealing with ambiguous pattern.

[1]Demonstrator is available at jedi.textmining.tu-berlin.de
[2]Freebase is available at www.freebase.com

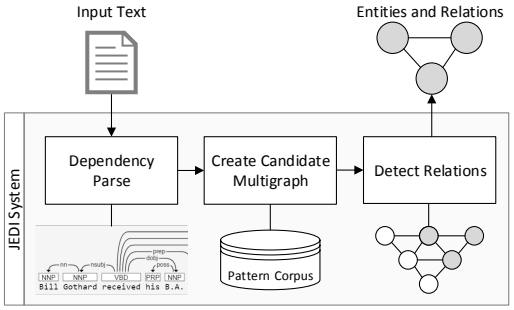



Figure 1: JEDI System Overview. Text is annotated with dependency information. Candidate entities connecting shortest path pattern are scored against a corpus. Constraint solving on the relation types resolves ambiguities and determines the final relations.

Consider the following sentences that both contain the pattern *receive* indicating completely different relations: "*Bill received his B.A. in Biblical Studies from Wheaton College.*", and "*Leo received an Oscar for Revenant.*".

The first sentence contains the binary relation *receiveDegree*, linking a person to a degree, but in the second, the same pattern indicates the *personAward* relation. To correctly disambiguate, we need to incorporate context. This paper proposes the novel method of using the entity types of multiple binary relations to solve the disambiguation.

**Motivation** Typically labeling relations and entities is done in sequence, leading to a pipeline architecture which first detects entities and subsequently tries to link them to extract relations. As detection errors can easily propagate, there is potential in executing these steps jointly. It is driven by the motivation that instead of focusing on individual binary assignments, multiple entities found in the text can be used to constraint and restrict each other's entity and relation types.

Figure 1 shows an overview of the entire solution. Entity mentions and potential relations indi-

cated by a pattern form a multi graph. Selecting a particular type for an entity prunes the number of potential relations that this entity can participate in. Furthermore, having established a relation between a pair of entities further restricts the potential relations that can hold between any other entity and one of the pair's members due to inferred type restrictions. Thus all candidate entities in a sentence are considered together to support or restrict any contained relations.

**Contributions** This paper presents and evaluates JEDI a system to translate the relation detection problem into a constraint satisfaction problem. The graph of entities and potential relations forms a resolution graph, where each entity is constrained on the potential types it can hold. Solving this problem jointly resolves entity and relation types without training an entity classifier, allowing to detect the large number of relation and types defined in FREEBASE.

The evaluation shows high precision across a variety of texts and relations. We furthermore release the implementation as open source and provide a web demonstrator to showcase the system.

## 2 Related Work

Most prominent for relation extraction is the idea that entities that co-occur with a similar context have similar meanings, driven by the distributional hypothesis (Harris, 1954). The shortest path kernel is a good estimator for approximating the distribution and has been used by Bunescu and Mooney (2005). Culotta and Sorensen (2004) showed that it is possible to train a relation classifier on the extracted pattern to predict a small number of relations.

Kate and Mooney (2010) proposed a card style resolution algorithm, which infers recursively the most probable relation assignment for a given entity pair, but still requires an entity classifier and only works for a small number of relations.

Mintz et al. (2009) proposed to use clustering to group together entity co-occurrences based on their shortest path, to extract relation types. This eliminates the need for a classifier for relation detection, but requires one for entity extraction. Pattern can only be assigned to one relation and thus capture only the most dominating meaning. The problem of relation extraction can also be solved using matrix decomposition, as shown by Riedel et al. (2013). Their work targets FREEBASE relations, but demands a complex training step which decomposes the co-occurrence matrix and is dependent on the text domain as well.

The SOFIE system (Suchanek et al., 2009) uses logical reasoning to find instances of a relation in text. It does not require any pre-training as it learns the extraction rules alongside the relation detection, but is limited in the amount of data it can process, because of the costly resolution step. Similar to our approach Singh et al. (2013) proposed to model the entity tagging and relation extraction problem jointly, using belief propagation to find relation instances, but targeting a much smaller number of relations. The idea to incorporate types into the relation extraction process was explored by Koch et al. (2014) improving the relation detection performance.

Contrary to existing systems JEDI does not need a pre-trained entity classifier. We leverage a very large corpus of shortest path patterns as reference and use constraint propagation to solve ambiguities. Our system also maps into the large number of predefined FREEBASE relations, alleviating the need to manually specify any relations. The system can be easily incorporated into a more complex IE pipeline that uses the results for example for entity linking.

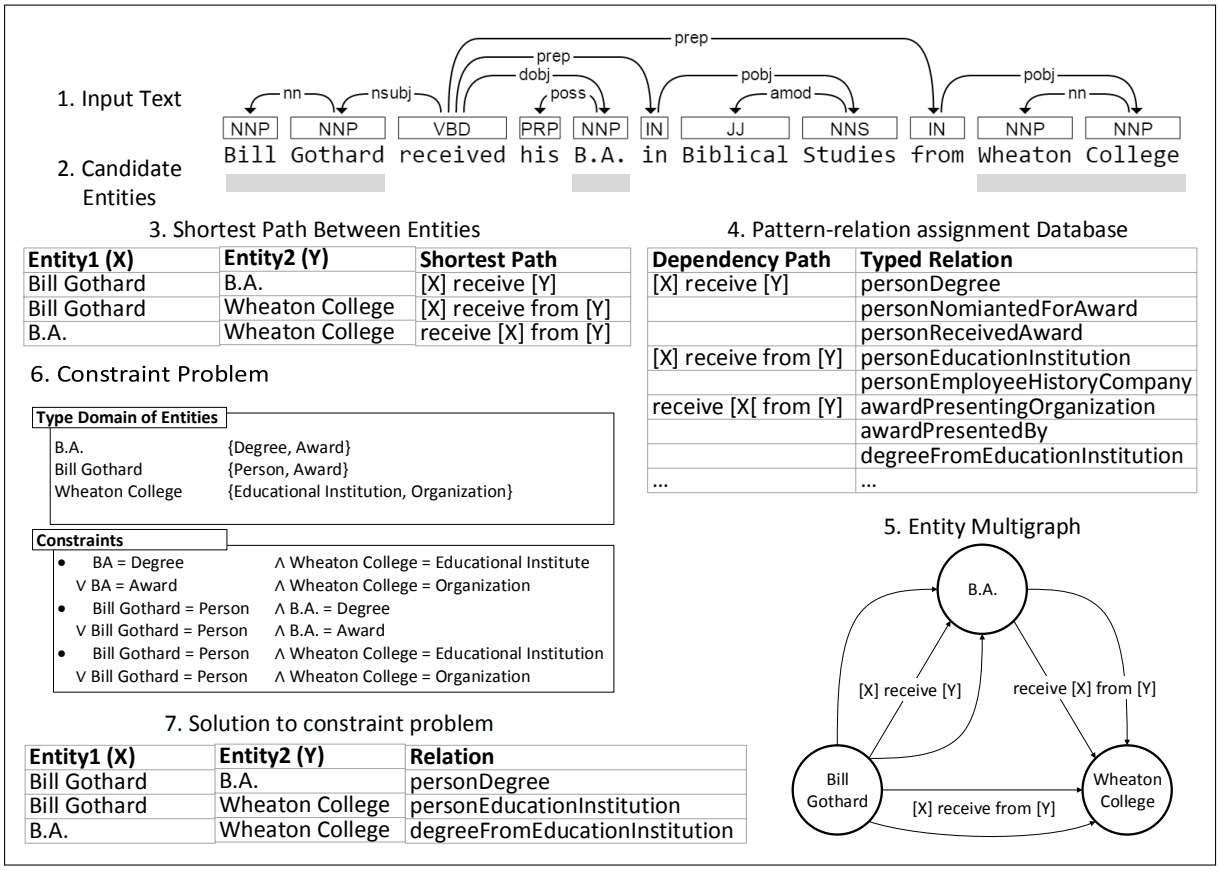
## 3 Pattern Corpus

To detect meaningful patterns we use FREEPAL (Kirschnick et al., 2014). A dataset of 10 million lexico-syntactic pattern-relation assignments built using distant supervision (Mintz et al., 2009). Each pattern was generated from a shortest dependency path between two known FREEBASE entities in a sentence that participate in a known relation. The corpus uses CLUEWEB09 as text basis and the FACC1 entity annotations (Gabrilovich et al., 2013) to generate a distribution of relations over pattern. An entropy score indicates the degree of ambiguity, which we use for scoring the relation assignments. Overall more than $75\%$ of the contained pattern were observed with more than one relation, requiring a disambiguation method.

## 4 Jointly Detecting Entities and Relations

The process of detecting relations is described in Figure 2 and consists of the following steps, described in the following:

- Pre-process input text




| Entity1 (X) | Entity2 (Y) | Shortest Path |
|---|---|---|
| Bill Gothard | B.A. | [X] receive [Y] |
| Bill Gothard | Wheaton College | [X] receive from [Y] |
| B.A. | Wheaton College | receive [X] from [Y] |

| Dependency Path | Typed Relation |
|---|---|
| [X] receive [Y] | personDegree |
| | personNomiantedForAward |
| | personReceivedAward |
| [X] receive from [Y] | personEducationInstitution |
| | personEmployeeHistoryCompany |
| receive [X[ from [Y] | awardPresentingOrganization |
| | awardPresentedBy |
| | degreeFromEducationInstitution |
| ... | ... |

| Entity1 (X) | Entity2 (Y) | Relation |
|---|---|---|
| Bill Gothard | B.A. | personDegree |
| Bill Gothard | Wheaton College | personEducationInstitution |
| B.A. | Wheaton College | degreeFromEducationInstitution |

Figure 2: Solution overview: Candidate Entities (2) are selected from the source text (1). Shortest path in the dependency tree is extracted (3), pruned against pattern-relation assignment database (4), type information is translated into a multi graph (5) which defines the constraint satisfaction problem (6). The solution yields an assignment of entity types and relations (7). (Types are omitted for readability)

- Selection of candidate entities
- Extract shortest dependency path (pattern) between all pairwise candidate entities
- Match the pattern using the FREEPAL corpus to determine candidate relations
- Translate the relation detection into a constraint satisfaction problem which determines the potential types of all entities and thus the connecting relations

### 4.1 Pre-Processing

The target text is annotated with part-of-speech tags and dependency information using the Stanford CoreNLP Toolkit (Manning et al., 2014). Coreference resolution is applied to further link entity mentions across sentence boundaries providing more link targets between entities.

### 4.2 Selecting Candidate Entities

Instead of trying to find any of the 10 million pattern from the pattern corpus in a given text, where every match would provide a candidate subject and object pair for a relation, we reverse the problem and produce a set of candidate entities and try to match the connecting pattern with the corpus.

JEDI works with any candidate entities, produced for example by an existing entity tagger or just based on simple heuristics. One such simple heuristic is to use nouns, with the extension to join together adjacent nouns to effectively form noun phrases. Nouns are grouped, if they are directly adjacent and connected through a dependency link of the type *poss* or *nsubj*, while also allowing the connecting word "of". This captures entities of the form "University of Illinois" and "Wheaton College", but fails to separate appositions such as "Bishop Oldham" or "Professor Smith", but this can be later rectified. This heuristic can be easily changed as the remaining processing does not depend on the text form or type of the entities. Using nouns also helps in finding entities generally not covered by specific NER systems, such as "biblical studies" as a field of study, without specifically training a tagger for this target type.

### 4.3 Extracting the Shortest Path

Finding the shortest path between two entities equals finding a path in the dependency graph between the head words of each entity. We use a simplified instance of Collins Head finding rule (Collins, 2003) to determine the head in multi-word entities. The pattern is derived by picking up all tokens in their lemmatized form on the path, substituting the start with X and the end with the label Y. To make the pattern more readable, all tokens are sorted based on their appearance in the source text.

This produces pattern of the form **[X] receive [Y] [1-dobj-2,1-nsubj-0]**. The pattern is further enriched with the individual dependency annotations to differentiate similar textual pattern[3].

**Conjunctions** We apply a simple normalization to conjunctions inspired by CLAUSIE (Corro and Gemulla, 2013). Removing the last token in patterns containing a conjunction dependency.

**Coreference Resolution** Coreference information expands mentions across sentence boundaries. Mentions that are connected through a chain are treated as if they are the same entity, if the source of the chain is marked as a candidate entity. Thus we substitute the coreference target with the source in the extraction process.

### 4.4 Pattern-Relation Assignments

The shortest path generation process generates a large number of pattern. To reduce the search space, all extracted pattern are matched against the FREEPAL corpus. This produces for each match a list of potential relations that this pattern has been observed with. Only pattern with an entropy smaller than *3.7*[4] and that have been observed at least five times are considered. This reduces the noise by filtering out very unspecific pattern such as *[X] be [Y]*, but at the same time still allows for a lot of ambiguous pattern.

Each pattern is associated with a list of FREEBASE relations, for which the argument types are retrieved. This is used to restrict the *X* and *Y* entity types of the pattern respectively. We use the FREEBASE type normalization presented in FIGER (Ling and Weld, 2012) to reduce the number of entity types down to 112.

[3]Dependency information for all pattern is omitted in the paper for readability, but used during the resolution process.

[4]This entropy cutoff was derived empirically.

To address the problem of arbitrary granularity, we broaden the accepted argument types using a simple type hierarchy. For example, the *diedIn* relation, which indicates that a person died in a particular location, restricts the subject argument to be of type deceased person. While this is very specific it prevents linking to this entity in other relations, which only accept the more generic person type. The type hierarchy is generated by retrieving the *type hints* category for each type, using the FREEBASE API. While this does not produce a complete type hierarchy, it adds the most commonly used sub types for a given type.

### 4.5 Constraint Solving using Type Inference

The extracted pattern for each pair of entities form a multi-graph, where edges are assigned a confidence score based on the FREEPAL entropy. The resolution process tries to eagerly generate a type assignment for each entity, so that at least one edge between connected vertices, a particular relation, holds according to the type requirements. The choco library (Prud'homme et al., 2015) is used for constraint solving. Each edge is transformed into a constraint, using logical conjunctions between all connected vertex pairs and disjunction for each edge between two vertices and their types. This emits for each relation a constraint with all possible type and subtype combinations.

**Scoring** Constraint solving produces more than one potential solution. We use a scoring mechanism to rank the different solutions, taking into account the number of matched entities, the entropy score taken from the FREEPAL dataset, as well as the type hierarchy. This ensures that if possible, the most specific type assignment for a large number of entities is favored in the resolution process.

**Backtracking** If there is no assignment possible - there is a conflict in the graph. Conflicts can arise when detecting relations that are not part of the corpus for a given pattern or wrong pattern as a result of erroneously linking entities in a co-reference chain. Backtracking is used to repeatedly remove vertices and all associated edges from the graph until either a solution is found or all nodes are removed in which case there is no solution. To find the highest scoring assignment backtracking is used to evaluate multiple different graphs, even when a solution is found.

| Relation | P | R | F |
|---|---|---|---|
| Education Degree | 0 | 0 | 0 |
| Place of Birth | 0.76 | 0.60 | 0.68 |
| Place of Death | 0.89 | 0.27 | 0.41 |
| Student Graduate | 0.78 | 0.41 | 0.53 |

Table 1: Baseline performance. Precision, Recall and F-measure without Coreference Resolution and type inference using the Noun strategy.

**Stopping** The search is terminated early when an adjustable time limit is hit, to ensure that the most probable solution is found early. This trade-off guarantees that the algorithm finishes in finite time, at the expense of not always finding the global optimal solution.

**Result** Once a solution is found, all vertices of the graph are bound to a type. The qualifying relation between any two connected vertices is selected as the one which has the highest score associated with it. This produces triples of the form <**entity, entity, relation**>.

## 5 Evaluation

The "Relation Extraction Corpus"[5] is used for evaluation – comparing precision, recall and F-measure. The corpus contains text snippets for four separate relations: person holding an education degree ***educationDegree*** (1580 triples), **place of death** (1955 triples), **place of birth** (8703 triples) and person graduated from education institute ***studentGraduate*** (32653 triples). Each excerpt is annotated by humans as to whether it supports a particular binary relation.

### 5.1 Results

**Baseline** Table 1 shows the baseline performance of the noun strategy without constraint solving, using the most likely relation for an identified pattern based on the FREEPAL entropy score.

The results show high precision for all relations except for education degree. This supports the use of the FREEPAL dataset for relation extraction, as it covers already a large variety of pattern instances. No instances of the education degree relation were found as almost all pattern for this relation are dominated by the received award relation.

[5]https://code.google.com/p/relation-extraction-corpus/

| Relation | P | R | F |
|---|---|---|---|
| Education Degree | 0.94 | 0.61 | 0.74 |
| Place of Birth | 0.77 | 0.60 | 0.67 |
| Place of Death | 0.88 | 0.35 | 0.50 |
| Student Graduate | 0.76 | 0.37 | 0.50 |

Table 2: Type inference performance without Coreference Resolution using the Noun strategy.

| | Resolution Strategies and Comparison Results | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Named Entities | | | Nouns | | | Akbik (2014) | | |
| Relation | P | R | F | P | R | F | P | R | F |
| Education Degree | **0.96** | 0.03 | 0.05 | **0.96** | 0.74 | 0.83 | 0.87 | 0.29 | 0.44 |
| Place of Birth | 0.77 | 0.52 | 0.62 | **0.83** | 0.58 | 0.68 | 0.82 | 0.19 | 0.31 |
| Place of Death | **0.92** | 0.48 | 0.63 | **0.92** | 0.48 | 0.63 | 0.82 | 0.13 | 0.22 |
| Student Graduate | 0.78 | 0.51 | 0.62 | 0.77 | 0.51 | 0.61 | **0.92** | 0.17 | 0.29 |

Table 3: Performance with type inference and Coreference Resolution using Named Entities and Nouns as entity markers, comparing to Akbik (2014), reporting Precision, Recall and F-measures.

**Type Inference** The effect of type inference can be seen in Table 2. Instances of the degree relation are found with high precision, while the other relations are still found, highlighting that the resolution process is not introducing errors.

Table 3 shows the performance of the entity candidate selection strategy (See section 4.2), including Coreference Resolution. The Noun strategy is compared with Named Entities obtained with the Stanford NER classifier (Finkel et al., 2005). For reference we present the results from Akbik et al. (2014), where the authors used a manual process to find a set of extraction pattern for a given relation.

Using the simple NOUNS strategy for selecting candidate entities performs on par or better to the NER strategy. Coreferences further improve the F-measure by up to .11 points. This supports our claim that we can detect entities and relations together, alleviating the need to train a custom classifier for an entity type - nouns are sufficient. The education degree relation (not part of the results) is a good example for an entity type, where the standard NER tagger almost always fails to identify the degree entity.

## 6 Conclusion

We present JEDI a system to extract typed named entities and FREEBASE relations together. Shortest dependency pattern are used to link entities, and constraint solving on the relation argument

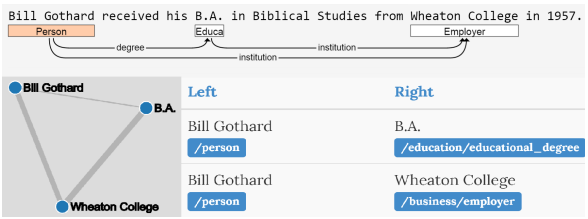



Figure 3: Demo system showing output of the relation detection process with found entity types and connecting FREEBASE relations.

types is used to disambiguate pattern with multiple meanings.

The evaluation shows that the method increases the precision and recall scores for ambiguous relations significantly. As the resolution takes advantage of entities that are connected in chains, it is further possible to detect n-ary relations using only binary pattern. The method proves to work well without any pre-training of NER classifiers and validates that pattern learned using distant supervision are effective. This makes it possible to expand existing knowledge bases with information found in web text.

A web demonstrator of the complete system as shown in Figure 3 is available at jedi.textmining.tu-berlin.de. The system is implemented as an UIMA module such that it can be easily incorporated into existing IE pipelines, the source code is hosted at github.com/jkirsch/jedi.

## Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. Johannes Kirschnick received funding from the German Federal Ministry of Economics and Energy (BMWi) under grant agreement 01MD15007B (SD4M) and Holmer Hemsen under 01MD15010A (SDW).

## References

Alan Akbik, Thilo Michael, and Christoph Boden. 2014. Exploratory Relation Extraction in Large Text Corpora. In *COLING*, pages 2087–2096.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *HLT/EMNLP*, pages 724–731.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637.

Luciano Del Corro and Rainer Gemulla. 2013. ClausIE : Clause-Based Open Information Extraction. In *WWW*, pages 355–365.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. *ACL*, pages 423–429.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. ACL, pages 363–370.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0).

ZS Harris. 1954. Distributional structure. *Word*, pages 775–794.

Rohit J. Kate and Raymond J. Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. *CoNLL*, pages 203–212.

Johannes Kirschnick, Alan Akbik, and Holmer Hemsen. 2014. Freepal: A Large Collection of Deep Lexico-Syntactic Patterns for Relation Extraction. In *LREC*, pages 2071–2075.

Mitchell Koch, John Gilmer, Stephen Soderland, and Daniel S Weld. 2014. Type-Aware Distantly Supervised Relation Extraction with Linked Arguments. In *EMNLP*, pages 1891–1901.

Xiao Ling and DS Weld. 2012. Fine-Grained Entity Recognition. *AAAI*, pages 94–100.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *HLT-NAACL*, pages 777–782.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. ACL, pages 1003–1011.

Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca, 2015. *Choco Documentation*. Available at `http://www.choco-solver.org`.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation Extraction with Matrix Factorization and Universal Schemas. In *HLT-NAACL*, pages 74–84.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *AKBC*, pages 1–6.

Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. 2009. SOFIE: A Self-Organizing Framework for Information Extraction. In *WWW 2009*.