# *RelationFactory*: A Fast, Modular and Effective System for Knowledge Base Population

**Benjamin Roth**[†] **Tassilo Barth**[†] **Grzegorz Chrupała**[*] **Martin Gropp**[†] **Dietrich Klakow**[†]
[†]Spoken Language Systems, Saarland University, 66123 Saarbrücken, Germany
[*]Tilburg University, PO Box 90153, 5000 LE Tilburg, The Netherlands
[†]{beroth|tbarth|mgropp|dietrich.klakow}@lsv.uni-saarland.de
[*]g.chrupala@uvt.nl

## Abstract

We present *RelationFactory*, a highly effective open source relation extraction system based on shallow modeling techniques. *RelationFactory* emphasizes modularity, is easily configurable and uses a transparent pipelined approach.

The interactive demo allows the user to pose queries for which *RelationFactory* retrieves and analyses contexts that contain relational information about the query entity. Additionally, a recall error analysis component categorizes and illustrates cases in which the system missed a correct answer.

## 1 Introduction and Overview

Knowledge base population (KBP) is the task of finding relational information in large text corpora, and structuring and tabularizing that information in a knowledge base. Given an entity (e.g. of type `PERSON`) with an associated relational schema (a set of relations, e.g. `city_of_birth(PERSON, CITY)`, `schools_attended(PERSON, ORGANIZATION)`, `spouse(PERSON, PERSON)`), all relations about the entity that are expressed in a text corpus would be relevant, and the correct answers would have to be extracted.

The TAC KBP benchmarks[1] are an effort to formalize this task and give researchers in the field the opportunity to evaluate their algorithms on a set of currently 41 relations. In TAC KBP, the task and evaluation setup is established by well-defined information needs about query entities of types `PERSON` and `ORGANIZATION` (e.g. who is the `spouse` of a person, how many `employees` does an organization have). A perfect system would have to return all relevant information (and only this) contained in the text corpus. TAC KBP aims at giving a realistic picture of not only precision but also recall of relation extraction systems on big corpora, and is therefore an advancement over many other evaluations done for relation extraction that are often precision oriented (Suchanek et al., 2007) or restrict the gold key to answers from a fixed candidate set (Surdeanu et al., 2012) or to answers contained in a data base (Riedel et al., 2010). Similar to the classical TREC evaluation campaigns in document retrieval, TAC KBP aims at approaching a true recall estimate by pooling, i.e. merging the answers of a timed-out manual search with the answers of all participating systems. The pooled answers are then evaluated by human judges.

It is a big advantage of TAC KBP that the end-to-end setup (from the query, through retrieval of candidate contexts and judging whether a relation is expressed, to normalizing answers and putting them into a knowledge base) is realistic. At the same time, the task is very complex and may involve too much work overhead for researchers only interested in a particular step in relation extraction such as matching and disambiguation of entities, or judging relational contexts. We therefore introduce *RelationFactory*, a fast, modular and effective relation extraction system, to the research community as open source software.[2] *RelationFactory* is based on distantly supervised classifiers and patterns (Roth et al., 2013), and was top-ranked (out of 18 systems) in the TAC KBP 2013 English Slot-filling benchmark (Surdeanu, 2013).

In this demo, we give potential users the possibility to interact with the system and to get a feel for use cases, strengths and limitations of the current state of the art in knowledge base population.

[1] `http://www.nist.gov/tac/about/`

[2] `https://github.com/beroth/relationfactory`

The demo illustrates how *RelationFactory* arrives at its conclusions and where future potentials in relation extraction lie. We believe that *RelationFactory* provides an easy start for researchers interested in relation extraction, and we hope that it may serve as a baseline for new advances in knowledge base population.

## 2 System Philosophy and Design Principles

The design principles of *RelationFactory* conform to what is known as the *Unix philosophy*.[3] For *RelationFactory* this philosophy amounts to a set of modules that solve a certain step in the pipeline and can be run (and tested) independently of the other modules. For most modules, input and output formats are column-based text representations that can be conveniently processed with standard Linux tools for easy diagnostics or prototyping. Data representation is compact: the system is designed in a way that each module ideally outputs one new file. Because of modularization and simple input and output formats, *RelationFactory* allows for easy extensibility, e.g. for research that focuses solely on novel algorithms at the prediction stage.

The single modules are connected by a makefile that controls the data flow and allows for easy parallelization. *RelationFactory* is highly configurable: new relations can be added without changing any of the source code, only by changing configuration files and adding or training respective relational models.

Furthermore, *RelationFactory* is designed to be highly scalable: Thanks to feature hashing, large amounts of training data can be used in a memory-friendly way. Predicting relations in real-time is possible using shallow representations. Surface patterns, ngrams and skip-ngrams allow for highly accurate relational modeling (Roth et al., 2013), without incurring the cost of resource-intensive processing, such as parsing.

[3] One popular set of tenets (Gancarz, 2003) summarizes the *Unix philosophy* as:

1. Small is beautiful.
2. Make each program do one thing well.
3. Build a prototype as soon as possible.
4. Choose portability over efficiency.
5. Store data in flat text files.
6. Use software leverage to your advantage.
7. Use shell scripts to increase leverage and portability.
8. Avoid captive user interfaces.
9. Make every program a filter.

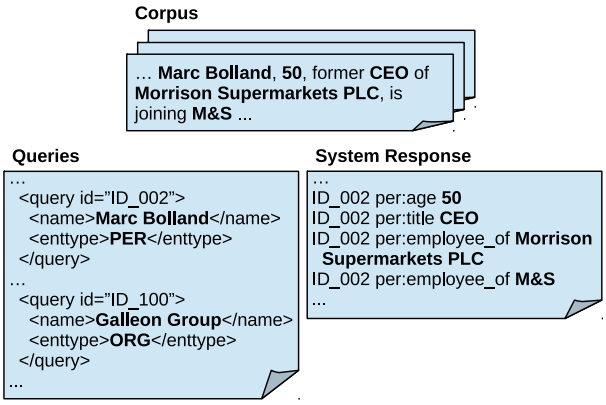


Figure 1: TAC KBP: Given a set of queries, return a correct, complete and non-redundant response with relevant information extracted from the text corpus.

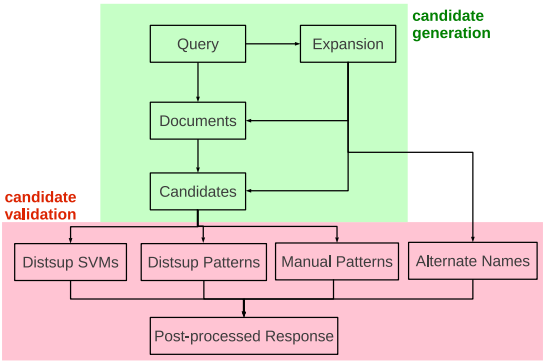


Figure 2: Data flow of the relation extraction system: The *candidate generation* stage retrieves possible relational contexts. The *candidate validation* stage predicts whether relations actually hold and produces a valid response.

## 3 Component Overview

A simplified input and output to *RelationFactory* is shown in Figure 1. In general, the pipeline is divided in a *candidate generation* stage, where documents are retrieved and candidate sentences are identified, and the *candidate validation* stage, which predicts and generates a response from the retrieved candidates (see Figure 2).

In a first step, the system generates aliases for the query using statistical and rule-based expansion methods, for example:

| Query | Expansion |
|---|---|
| Adam Gadahn | Azzam the American, Adam Yahiye Gadahn, Gadahn |
| STX Finland | Kvaerner Masa Yards, Aker Finnyards, STX Finland Ltd |

The expansions are used for retrieving documents from a Lucene index. All those sen-

tences are retained where the query (or one of the query aliases) is contained and the named-entity tagger has identified another entity with the type of a potential answer for one of the sought relations. The system is easily configurable to include matching of non-standard named-entity types from lists. *RelationFactory* uses lists obtained from Freebase (`www.freebase.com`) to match answer candidates for the types `CAUSE-OF-DEATH`, `JOB-TITLE`, `CRIMINAL-CHARGES` and `RELIGION`.

The candidate sentences are output line-by-line and processed by one of the *validation modules*, which determine whether actually one of the relations is expressed. *RelationFactory* currently uses three standard validation modules: One based on SVM classifiers, one based on automatically induced and scored patterns, and one based on manually crafted patterns. The validation modules function as a filter to the candidates file. They do not have to add a particular formatting or conform to other requirements of the KBP task such as establishing non-redundancy or finding the correct offsets in the text corpus. This is done by other modules in the pipeline, most notably in the post-processing step, where statistical methods and heuristics are applied to produce a well-formed TAC KBP response.

## 4 User Perspective

From a user perspective, running the system is as easy as calling:

```
./run.sh system.config
```

The configuration file contains all information about the general run configuration of the system, such as the query file to use, the format of the response file (e.g. TAC 2012 or TAC 2013 format), the run directory that will contain the response, and the Lucene index with the corpus. Optional configuration can control non-standard validation modules, and special low or high-recall query expansion schemes.

The relevant parts of the configuration file for a standard 2013 TAC KBP run would look like the following:

```
query /TAC_EVAL/2013/query.xml
goal response2013
rundir /TAC_RUNS/run2013/
index /TAC_CORPORA/2013/index
rellist /CFG/rellist2013
relations.config /CFG/relations2013.config
```

The last two lines refer to relation-specific configuration files: The list of relations to use and information about them. Changing these files (and adding respective models) allows for inclusion of further relations. The relation-specific configuration file contains information about the query entity type, the expected answer named-entity tag and whether a list of answers is expected (compared to relations with just one correct answer):

```
per:religion enttype PER
per:religion argtag RELIGION
per:religion listtype false
org:top_members_employees enttype ORG
org:top_members_employees argtag PERSON
org:top_members_employees listtype true
```

*RelationFactory* comes with batteries included: The models and configurations for TAC KBP 2013 work out-of-the-box and can easily be used as a relation extraction module in a bigger setting or as a baseline for new experiments.[4]

## 5 Illustrating *RelationFactory*

In TAC KBP 2013, 6 out of 18 systems achieved an F1 score of over 30%. *RelationFactory* as the top-performing system achieved 37.28% compared to 68.49% achieved by human control annotators (Surdeanu, 2013). These numbers clearly show that current systems have just gone halfway toward achieving human-like performance on an end-to-end relation extraction task.

The aim of the *RelationFactory* demo is to illustrate what the current challenges in TAC KBP are. The demonstration interface therefore not only shows the answers generated for populating a potential knowledge base, but also what text was used to justify the extraction.

The real-time performance of *RelationFactory* allows for trying arbitrary queries and changing the configuration files and immediately seeing the effects. Different expansion schemes, validation modules and patterns can be turned on and off, and intuitions can be obtained about the bottlenecks and error sources of relation extraction. The demo also allows for seeing the effect of extracting information from different corpora: a Wikipedia corpus and different TAC KBP corpora, such as newswire and web text.

[4] Training models for new relations requires is a bigger effort and includes generation of distant supervision training data by getting argument pairs from relational patterns or a knowledge base like Freebase. *RelationFactory* includes some training scripts but since they are typically run once only, they are significantly less documented.

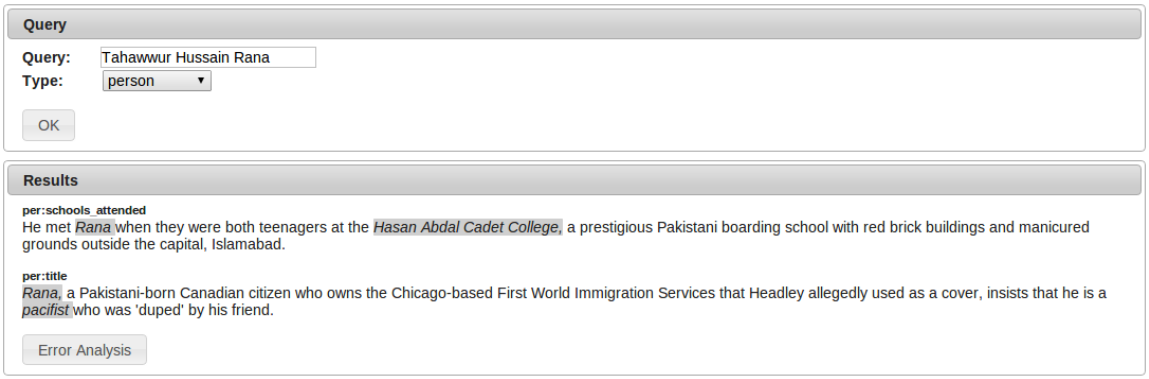


Figure 3: Screenshot of the *RelationFactory* demo user interface.

*RelationFactory* contains a number of diagnostic tools: With a gold key for a set of queries, error classes can be broken down and examples for certain error classes can be shown. For example, the diagnostic tool for missed recall performs the following checks:

1. **Is document retrieved?**
2. **Is query matched?** This determines whether a sentence is considered for further processing.
3. **Is answer in query sentence?** Whether the answer is in one of the sentences with the query. Our system only can find answers when this is the case, as there is no coreference module included.
4. **Do answer tags overlap with gold answer?**
5. **Do they overlap exactly?**
6. **Other (validation).** If all previous checks are passed, the candidate has correctly been generated by the candidate generation stage, but the validation modules have failed to predict the relation.

On the TAC KBP 2013 queries, the resulting recall error analysis is:

| error class | missing recall |
|---|---|
| Doc not retrieved | 5.59% |
| Query not matched | 10.37% |
| Answer not in query sentence | 16.63% |
| Answer tag inexact | 5.36% |
| Answer not tagged | 24.85% |
| Other (validation) | 37.17% |

The demonstration tool allows for inspection of instances of each of the error classes.

## 6 Conclusion

This paper illustrates *RelationFactory*, a modular open source knowledge-base population system. We believe that *RelationFactory* will become especially valuable for researchers in the field of relation extraction that focus on one particular problem of knowledge-base-population (such as entity expansion or relation prediction) and want to integrate their algorithms in an end-to-end setting.

## Acknowledgments

Benjamin Roth is a recipient of the Google Europe Fellowship in Natural Language Processing, and this research is supported in part by this Google Fellowship. Tassilo Barth was supported in part by IARPA contract number W911NF-12-C-0015.

## References

Mike Gancarz. 2003. *Linux and the Unix philosophy*. Digital Press.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh, and Dietrich Klakow. 2013. Effective slot filling based on shallow distant supervision methods. In *Proceedings of the Sixth Text Analysis Conference (TAC 2013)*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL)*, pages 455–465. ACL.

Mihai Surdeanu. 2013. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *Proceedings of the Sixth Text Analysis Conference (TAC 2013)*.