

# Classification Algorithms for Relation Prediction

Christoph Boden<sup>#1</sup>, Thomas Häfele<sup>#2</sup>, Alexander Löser<sup>#3</sup>

<sup>#</sup> *University of Technology Berlin  
Einsteinufer 17, 10587 Berlin, Germany*

<sup>1</sup>christoph.boden@campus.tu-berlin.de

<sup>2</sup>thomas.haefele@campus.tu-berlin.de

<sup>3</sup>alexander.loeser@tu-berlin.de

*Abstract*— Knowledge discovery from the Web is a cyclic process. In this paper we focus on the important part of transforming unstructured information from Web pages into structured relations. Relation extraction systems capture information from natural language text on Web pages, called Web text. However, extraction is quite costly and time consuming. Worse, many Web pages may not contain a textual representation of a relation that the extractor can capture. As a result many irrelevant pages are processed by relation extractors.

We propose a relation predictor to filter out irrelevant pages and substantially speed up the overall information extraction process. As a classifier, we trained a support vector machine (SVM). We evaluate pages on a sentence level, where each sentence is transformed into a token representation of shallow text features.

We evaluate our relation predictor on 18 different relation extractors. Extractors vary in their number of attributes and their extraction domain. Our evaluation corpus contains more than six million sentences from several hundred thousand pages. We report a prediction time of tens of milliseconds per page and observe high recall across domains.

Our experimental study shows that the relation predictor effectively forwards only relevant pages to the relation extractor. We report a speedup of at least factor two while discarding only a minimal amount of relations. If only fixed amount e.g. 10% of the pages in the corpus are processed, the predictor drastically increases the recall by a factor of five on average.

## I. INTRODUCTION

Knowledge discovery from the Web is a cyclic process. First, relevant data sources have to be identified and tapped. Second, the data has to be extracted and formalized. Third, data has to be linked and connected and finally will be used in various applications, e.g. to populate knowledge bases that can be used to answer complex analytical questions. Finally, the user feedback may trigger this process from scratch. For instance, a user may discover that some relations are missing. In that case the user may retrigger the entire process with a new search for additional data sources (see also Fig 1).

In this paper we focus on the important part of formalizing information in this life cycle. For instance, formalizing information from Web pages is accomplished through information extraction and reasoning, which includes recognizing named entities and extracting semantic relations between them.

Current relation extraction systems, such as [6][5][2], utilize a computational expensive execution stack of part-of-speech tagging, phrase detection, named entity recognition and co-reference resolution. This machinery is needed to

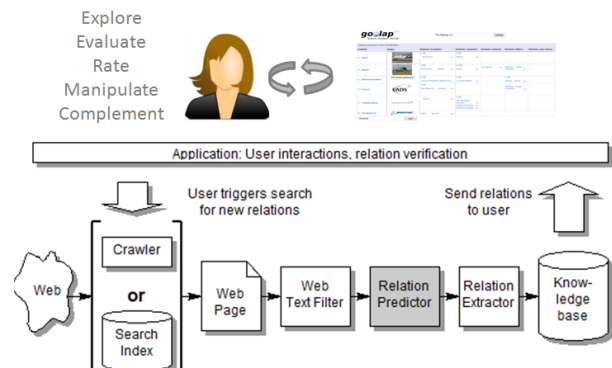


Fig 1: Relation extraction pipeline with our relation prediction component.

detect the exact borders of attribute values, to identify their correct type and to identify their relation within and across sentences. As a result, processing a Web page may take up to a few seconds [10]. Higher precision is ensured with deep dependency parsing or semantic role analysis. These approaches require processing times of tens of seconds [10]. Recently, extraction system vendors also offer extraction-as-a-service [5]. These systems enable non-natural-language-processing experts to extract relations from Web text. This abstraction comes at the price that users can no longer tune the patterns or rules of the extractor. Moreover, additional costs may appear, such as costs for sending a page through the network or monetary costs.

To obtain relevant Web pages one can either apply focused crawling on the web or query existing web search engines with generated keywords [7]. But even with these focused approaches only a small fraction of the retrieved web pages actually contains a textual representation of a semantic relation that the extractor can capture (See Figure 2). It would thus be highly desirable to be able to filter out irrelevant pages that do not contain relations to speed up the extraction process and only forward relevant pages to the relation extractor.

**Our Contribution:** We propose a component called relation predictor that predicts whether an extraction service will be able to extract a relation from a web Page. The relation predictor only forwards relevant pages which contain a textual representation of a relation to the actual extraction service. The entire extraction pipeline is shown in Fig 1.

The relation predictor abstracts from extractor specific words and syntactic structures and is generally applicable to

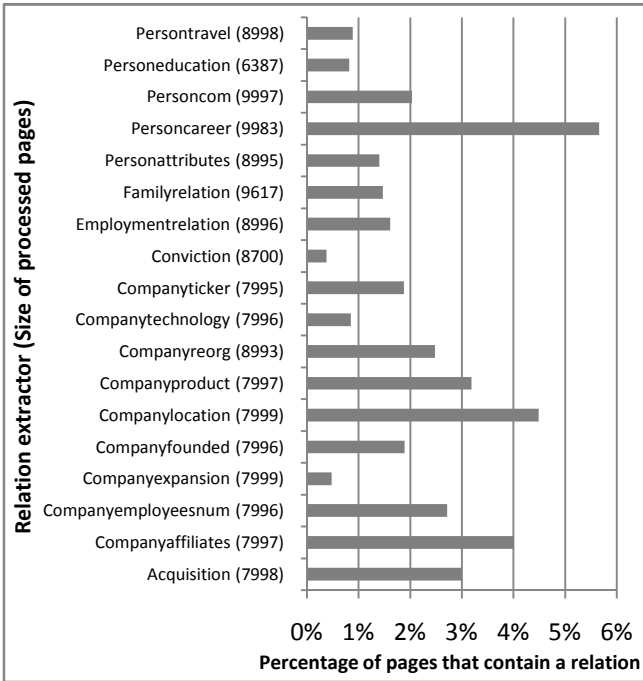


Fig 2: This Figure shows the percentage of forwarded pages that contain a relation for 18 different relation extractors. Only a small fraction of processed pages de facto contains a relation.

multiple extraction services. It avoids computationally intensive natural language processing techniques, such as part-of-speech tagging or deep dependency tree parsing.

## II. DATA SET

Relation extraction involves computationally expensive operations for each processed page. Therefore an important problem is forwarding only relevant pages to a relation extractor.

### A. Training and Evaluation Dataset

To be able to consistently train and evaluate a predictor, we decided to generate a fixed corpus of annotated web Pages. We choose 18 different relationship-type of the domains company and person from a public extraction service [5]. For each relationship-type we generated relation-specific keyword queries with a keyword generation strategy [7]. For each type of relation we queried a Web search engine service[4] with these queries and retrieved the top-1000 . Some pages could not be downloaded however, so overall we retrieved about 153.000 pages. Each page was also forwarded to a the relation extraction service OpenCalais[5] to annotate each page with the relations it contains. This annotation is seen as the gold standard, since we aim to predict whether a given relation extractor will be able to extract a relation.

### B. Only a fraction of forwarded pages contain a relation

Figure 2 shows the distribution of relations for different relationship types. Note, that because of the generated

keywords, this sample is already optimally biased towards pages that likely contain a specific relation. Nevertheless we observe a rather sparse distribution of relations in our corpus. Apparently, extraction systems retrieve many irrelevant pages that do not contain any text that represents a relation.

## III. RELATION PREDICTION

The problem of detecting a relation in a Web page is essentially a binary text classification problem. A classifier generally consists of a feature extraction component that extracts numeric values relevant to the decision and the actual classification algorithm that predicts the label of a document given the features.

In this section we describe the general system setup, the linguistic feature extraction process and the actual classification algorithm we use.

### A. Experimental Setup

Figure 3 shows the two different extraction pipelines that we use to train and evaluate the relation predictor. Once Web Pages have been retrieved either by crawling or via a search engine, they are sent through Web Text Filter. This component removes navigational elements, templates, and advertisements (so called boilerplate). It only keeps sentences and paragraphs [11] of a page. Figure 3 shows the pipeline from this point on.

1) *Training the Predictor*: Most semantic relations are expressed within a single sentence. To honour that fact, we choose to evaluate each Web page on a sentence level rather than treating the whole page as a bag of words. To generate labeled training instances for the classifier, we use a setup depicted in Figure 3 (upper half). First, we forward training pages to a relation extractor, which labels sentences as either containing a specific relation or not. Next, we transform each sentence to a token representation of shallow text features as described in Section III.B.

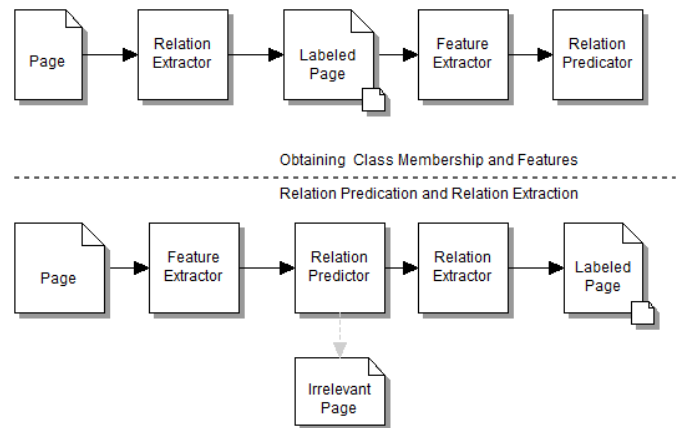


Fig. 3: Experimental Setting: The upper pipeline shows the setup for training the predictor. The lower pipeline illustrates the setup used during prediction.

Feature Class	Token	Encoding	Implementation	Example	Role for Predicting a Relation	Explanation
Shallow	starts with lower case token	L	shallow text	wood stock (LL), green apple (LL)	indicates attribute value, indicates relationship	indicates lower cased part of speech elements, such as adverbs, adjectives, verbs, lower cased nouns, particles, interjections etc.
	starts with upper case token	U	shallow text	NBA (U), Mark Hurd (UU)	indicates attribute value	indicates proper nouns
	starts with digit	D	shallow text	1997, 25th., 50%, 80legs.com	indicates attribute value	indicates numerical value or values that represent a date or time
	starts with other punctuation	!	shallow text	?,!, ;	sentence structure	separates clauses in sentence
Syntactic	is determiner (article, quantifiers, universal determiners only)	T	hash list	a, an, the, all, few, many, several, some, every, each, any, no, all, both	indicates attribute value	is a noun-modifier, connects noun or noun-phrase to context of sentence
	is comma	,	shallow text	Barack Obama , Angela Merkel	indicates attribute value	represents a list of values
	is dash	—	shallow text	Flight Paris — London	indicates a relationship	represents a closed range relation between values or a contrast between values
	is semicolon	;	shallow text	Peter went out; he was back at 5 am.	sentence structure	separates clauses in sentence
	is coordinating conjunction that indicates a sequence	C	hash list	pizza or pasta, Ernie and Bert, Mark, Peter & Paul	indicates attribute value	represent a list of values
	is preposition	P <sub>p</sub>	hash list	of, at, to, instead of, prior to, in spite of	indicates a relationship	links attribute value to other tokens, such as predicates, verbs and nouns
	ends with possessive apostrophe	'	shallow text	IBM's DB2, Inga's daughter Mathilde	indicates a relationship	represents possessive relation
	is colon	:	shallow text	Catherine Zeta-Jones Filmography: High	indicates a relationship	predicate of a relation, substitutes verb phrase
	is personal pronoun	R <sub>PP</sub>	hash list	I, you, we	indicates attribute value	represents initiator of relation, substitutes for proper or common nouns
is objective pronoun	R <sub>OBJ</sub>	hash list	me, him, her,	indicates attribute value	represents target of a verb that indicates the relation	
is possessive pronoun	R <sub>PSS</sub>	hash list	mine, yours, his, hers, its, ours, theirs	indicates attribute value	represents possessive relation	
Semantic	is relation extractor-specific verb phrase	V	hash list	BOBJ was acquired by SAP (i.e. for relation extractor 'acquisition')	indicates a relationship	represent a verb / prepositional verb phrase in a relation

Fig 4: Our tag set contains 16 different tag classes used for predicting relations. Each sentence is tokenized and transformed into a general representation using only these 16 tags. Tags help to identify tokens that represent structure in sentences, tokens that represent potential attribute values or tokens that represent a relation for a specific type.

Finally we encode the occurrence of bigrams of these tokens as binary features  $x_i$  to produce  $l$  labeled training samples of type  $(x_i, y_i)$  ( $1 \leq i \leq l$ ), where  $x_i$  is the feature vector in our  $n$  dimensional feature space, and  $y_i \in \{-1, +1\}$  denotes the class label.

2) *Prediction*: At prediction time we segment a page into sentences and transform each sentence into its feature vector representation. We then predict the label for each sentence in the document. If a single sentence in a page is predicted as positive, the entire page is forwarded to the relation extractor. Otherwise, the page is discarded and removed from the relation extraction pipeline. Figure 3 (lower half) illustrates this process.

### B. Linguistic Feature Analysis

Predicting relations for thousands of relation extractors is difficult. For instance, word-based n-gram models can result in tens of thousands of relevant features, which apparently would make the relation predictor susceptible to over fitting to a particular page subset. Therefore authors of [2] utilized part-of-speech tags and predefined patterns for predicting binary relations. Because of the part-of-speech analysis this approach is computational expensive. Furthermore it is only limited to binary relations, in particular relations in which entities are separated by few tokens only. Authors of [12] overcome this shortcoming and recognize the structural locality of generic relations with deep dependency tree parsing.

1) *Relation prediction requires 16 tags only*: We transform each sentence into a set of tokens representing shallow text features. Figure 4 gives an overview our set of 16 token tags.

In the reminder of this section we illustrate our principles for extracting lexical, syntactic and semantic features of attribute values and relations.

2) *Sentence Transformation*: General attribute values as shallow text features. Authors of Web text may ‘invent’ new nouns to express an attribute value. Therefore the number of potential attribute values may become too large to be held in a hash list. We encode any lowercased tokens, such as verbs and nouns, as (L), tokens starting with an uppercase character, such as proper nouns or acronyms, as (U) and tokens starting with a digit as (D).

3) *Special Case: Closed Classes as Hash Lists*: Lower cased nouns may start with determiners, common nouns or pronouns. For these closed classes authors of Web text rarely invent new words. Closed classes help us distinguishing lower cased nouns from other lowercased tokens. We encode English language prepositions (P), determiners (T) and pronouns (R) as hash list.

4) *Special case: Comma*: Sequence of noun phrases, proper nouns or digits are a rich source of attribute values. We encode the comma as (,), and encode coordinating conjunctions (C) as hash list, such as ‘and’, ‘or’, ‘&’.

*Example 3.1*: The following examples showcase our encoding rules for typical phrases that may express an attribute value:

**Complex noun**: the wooden tool → TLL

**Proper noun:** *SAP AG* → *UU*

**Acronym:** *IBM* → *U*

**Complex proper noun:** *Sony VX 5a* → *UUD*

**Date example:** *25 September 1969* → *DUD*

5) *Extracting Relations: Verb phrases as hash list:* Verb phrases and prepositional verb phrases frequently indicate a relation between two entities in English language Web text [2]. For instance, prepositional verbs resemble phrasal verbs in that both verb forms consist of a verb followed by a preposition. Relation extractors utilize verb dictionaries to determine the semantic type of a relation. We reuse these dictionaries and generalize single and multiword verb phrases in sentences as (V). Thereby, we assume that a relation classifier can access these verb phrase dictionaries. For instance, dictionaries can be obtained from rules of declarative relation extractors [7].

For obtaining a representative hash list of verbal phrases, we bootstrap the relation extractor with a small sample of pages. Next we apply the method of [10] and observe prepositional verb phrases in returned pages. The method is efficient and requires only a few hundred pages.

6) *Special Case: Appositive and Possessive Relations as Shallow Text Features:* We add a tag (‘) for encoding relations that are expressed with a possessive apostrophe. We also add the colon tag (:) for encoding relations that are expressed as appositive phrase.

*Example 3.2: The following phrases showcase our encoding rules for typical relationship types. We underline the plain text and the corresponding features that express the relation:*

**Relation with verb phrase:**

*SAP AG recently acquired performance management vendor Cartesis. → UULVLLL\_(V<sub>ACQUISITION</sub>=‘acquired’)  
Relation Extractor: ACQUISITION (Acquirer, Acquired)*

**Relation with possessive apostrophe:**

*said Berlin’s politically ambitious mayor, Klaus Wowereit on a press conference yesterday. → LU’LLL,UUP<sub>P</sub>TLLL  
Relation Extractor: PERSON\_POSITION(Person, Position)*

7) *Feature Class Hierarchy:* Sometimes a token can be mapped to multiple encodings. Part-of-speech taggers disambiguate different part-of-speech classes within a single sentence. However, this high precision approach comes at the price of a large feature set that causes longer execution times. The focus of our scenario is different. We can relax the requirement of a precise interpretation for each sentence that would require a part-of-speech tagger. Instead, we introduce a hierarchy among different feature extractors. We consider shallow features as less discriminative than syntactic features and syntactic features as less discriminative than semantic features. If a token or a set of tokens matches for multiple feature class, we select the feature class that is most discriminative. For instance, the token ‘acquired by’ can be

encoded as LL, LPP or V. We select the encoding V since it belongs to the most discriminative feature class.

8) *Feature Extraction:* After a sentence is transformed into its token representation, we encode the occurrence of bigrams of these tokens as binary features  $x_i$  to generate the feature vectors to be sent to the classification algorithm.

*C. Classification Algorithm*

The problem of detecting a relation in a Web page is essentially a binary text classification problem. We employ linear Support Vector Machines (SVMs), which have already been shown to produce superior accuracy on related tasks such as text categorization [20]. SVMs are maximum margin classifiers based on the structural risk minimization principle [13]. SVMs can be trained efficiently as binary classifiers and generalize well on unseen data. Linear support vector machines find a hyper-plane that separates the classes of training examples with maximum margin. This can be formulated as a convex optimization problem. We used the *liblinear* implementation of a L2-loss-L2R soft margin SVM [14], which solves the following quadratic program:

$$\begin{aligned} \min_{w, \xi_i} & \frac{1}{2} \|w\|^2 + C \sum_i \xi_i^2 \\ \text{s. t.} & y_i(w * x_i + b) \geq 1 - \xi_i \quad \forall x_i \\ & \xi_i \geq 0 \end{aligned}$$

In this equation, C denotes the cost of misclassification and is to be determined through cross validation. The soft-margin SVM allows for some misclassifications of training samples through the slack variable  $\xi_i$ . To accommodate for the fact that our training data is heavily unbalanced (>99% of the sentences are negative training samples), we introduce different weights for the misclassification cost  $c_i$  for each class  $y_i \in \{-1, +1\}$ .

IV. EXPERIMENTAL EVALUATION

In this Section we report on the robustness, recall, costs, execution time and performance of our relation predictor.

We divided our corpus into two parts: 60% for training and 40% for testing the relation predictor. From these pages we removed boilerplate templates [11] and finally extracted 6.7 million sentences. From the test set we removed non English pages and pages that did not contain any extractable content. Overall our test set that we report our results on contains 42,000 pages. We trained the classifier for each relation predictor in our data set with different settings on the 60% training pages with different parameters. We then evaluated these classifiers on the test set.

We evaluated the predictors against a gold standard. It reflects the maximum number of relations that the evaluated relation extraction service could extract from our corpus, if all pages would have been processed. We ran two different scenarios:

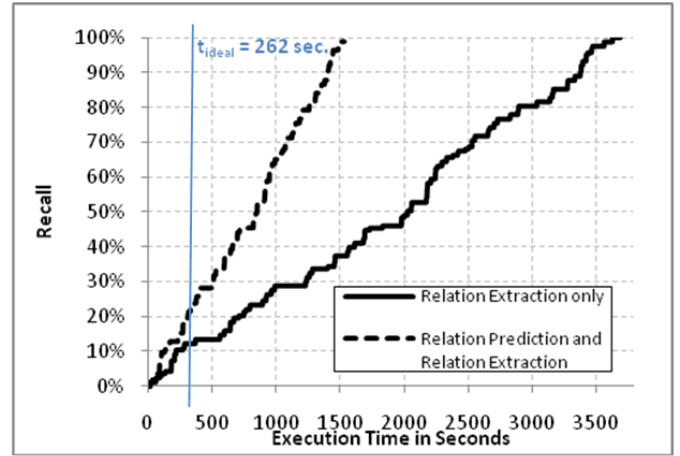
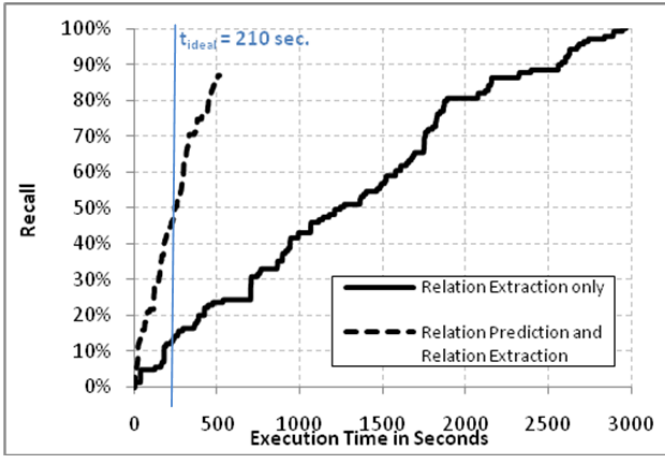


Fig 5: We measure execution time and recall for the relation extraction pipelines acquisition (left) and companyreorganization (right) with and without the predictor on the test data set. The predictor significantly speeds up the relation extraction process by at least a factor of 2 with only modest information loss. (Note that the Web pages were processed in the same order.)

**Relation Extraction only.** This scenario implements a full scan [3] over the entire set of pages in our test corpus. The scan is executed in a random order. During the scan, each page in the test corpus is forwarded to the relation extractor. Note that this scenario reflects the current access method for public extraction services, such as [5], or the access method of several academic relation extraction systems, such as [15][12][6][2][1].

**Relation Prediction + Relation Extraction.** This scenario utilizes our relation predictor. Each page in the test set is analysed by the relation predictor. For each page the predictor decides whether the page is forwarded to the relation extractor or discarded.

#### A. Performance Improvements

To demonstrate the performance impact of our predictor, we measure the execution time of the entire relation extraction pipeline both with and without the predictor component on the entire test corpus. Figure 3 (lower part) illustrates this process. Figure 5 shows the recall and execution time for two example relation extraction pipelines ‘acquisition’ and ‘company reorganization’.

Our relation predictor is about two orders of magnitude faster than the actual relation extraction service. (The prediction time includes boilerplate removal, sentence segmentation, feature extraction and the actual prediction computation.) For example the predictor takes 27ms per Web page on the relation ‘company location’ and 92ms per Web page on the relation ‘person career’.

Imagine a perfect predictor that only forwards relevant pages. The time it would take this predictor to process the entire corpus of Web pages is given by:

$$t_{ideal} = n_{pos\ documents} \times (t_{extraction} + t_{prediction}) + n_{neg\ documents} \times t_{prediction}$$

It represents the lower bound for the execution time. This time is represented by the two vertical lines in Figure 5. As an example, we show predictors suited for the ad-hoc query scenario. Here low execution time is the most crucial component, while some missed relations are acceptable.

For both presented relation extraction pipelines there is a substantial speedup in the system, when the predictor is introduced. In the case of ‘acquisition’ (left), the system with a predictor is about five times faster than the unfiltered system while still achieving almost 90% recall. Note this is only twice as long as the ideal predictor would take. In the case of ‘company reorganization’ (shown on the right) there is still a speedup of factor two, while almost 100% recall is achieved. We observed similar results for the other 16 relation extractors. However the numbers have been omitted due to the limited space.

#### B. Recall Enhancement across Domains

To illustrate the performance improvements for different types of relation extractors, we measured the recall (Percentage of total facts retrieved) after 10% of the pages in the test corpus have been processed by the relation extractor. The results for 18 different relation extractors are shown in Figure 6.

The ‘prediction + extraction’ pipeline achieves significantly higher recall than the pipeline without the prediction component across almost all different relation extractors. This is a remarkable improvement.

#### C. Discussion

Our evaluation results clearly show the usefulness of our relation predictor. The relation predictor robustly predicts relevant pages for 18 different relation extractors.

1) *Robust Prediction for a Wide Range of Attribute and Relationship Types:* With our simple and small feature set the predictor recognizes the textual representation of values for very different attribute types, such as date, company, person,

## V. RELATED WORK

In this Section we review work on text filtering, pattern generation and relation extraction optimization.

### A. Text Filtering and Pattern Learning

1) *Text Filtering*: Most closely related to our approach is the usage of information extraction systems to perform text-filtering, as discussed in MUC-4 studies [17]. Text filtering components at MUC-4 pre-label sentences with named entities or part-of speech information (or expect pre-labeled documents). Next, a rule-based pattern or a statistical classifier filters out documents that do not belong to a specific relationship type.

2) *Pattern Learning*: Riloff [16] pioneered a learning procedure in which documents are marked as relevant to the relation extraction task. The learning procedure receives a set of POS-tagged and outputs a set of textual patterns that can be used for discovering additional relations. A plethora of different pattern learning system enhanced this approach (see [9] for an overview). For instance, in the PROTEUS System [18], authors used a set of regular expressions pattern for learning good-quality patterns automatically from a large, general, un-annotated corpus of documents. Most recently, systems for Open Information Extraction (OIE) [2][15] extract short (up to 5 token) and generic relations between noun phrases in English language Web text. These generic relations are extracted with few syntactic patterns on POS labeled text. Later, the same authors [10] proposed preliminary ideas to capture also distant relations with the help of deep dependency parsers.

Our relation prediction approach is fundamentally different from previous text filtering and pattern generation approaches. We do not require tags generated through part-of-speech tagging, noun phrase chunking, named entity recognition, coreference reconciliation or deep parsing. As a result, we predict relations in a page in a few milliseconds, instead of seconds (with part-of-speech tagging) or tens of seconds (with deep parsing) [17]. Furthermore our schema is generally applicable to a wide range of relation extractors and not fine-tuned to individual scenarios such as in the MUC-4.

We assume a relation extraction service as a black box. Moreover, and contrary to existing pattern generation approaches, we do not require sending feedback to the relation extractor, making our approach applicable for a broad range of relation extraction scenarios on the Web.

Unfortunately, MUC-4 studies could not verify the usefulness of text filters for improving the performance of a relation extraction system. To our best knowledge, we are the first study that shows the general applicability and usefulness of predicting relations for a wide range of different relation extraction systems. Our results demonstrate a drastic reduction of processed pages while remaining a high recall.

Relation Extractor	Fact Recall after 10% processed pages	
	Extraction Only	Prediction + Extraction
acquisition	16.5%	70.5%
companyaffiliates	9.7%	64.8%
companyemployeesnumber	11.2%	43.8%
companyexpansion	6.3%	25.0%
companyfounded	10.1%	60.9%
companylocation	13.9%	42.4%
companyproduct	10.9%	51.7%
companyreorganization	13.5%	84.0%
companytechnology	10.0%	43.3%
companyticker	11.5%	93.4%
conviction	0.0%	42.1%
employmentrelation	3.9%	66.7%
familyrelation	17.2%	17.2%
personattributes	7.3%	41.8%
personcareer	8.9%	12.8%
personcommunication	9.2%	19.5%
personeducation	0.0%	57.1%
persontravel	14.3%	25.7%

Fig 6: The figure shows the Recall after 10% of the pages in the test corpus have been processed by the relation extractor both with and without the prediction component in the pipeline. The relation predictor drastically increases recall by effectively filtering out irrelevant pages across various types of relations.

location, product, product type, technology, conviction charge, ticker symbol, job position, or degree. Moreover, this feature set enables the relation predictor not only to recognize binary relations, but also complex relations with five or more attribute values in Web text.

2) *Robust Prediction for rare and frequent Relationship Types*: The relation predictor robustly identifies relevant pages for both, frequent and infrequent, relationship types. As a result, our relation predictor is particularly helpful for ‘rare’ relationship types. For instance, relations of the type ‘*company technology*’, ‘*conviction*’, ‘*company expansion*’, or ‘*person education*’ appear in less than 2% of the pages. Our relation predictor effectively filters out irrelevant pages for these relationship types.

3) *Slightly higher Recall for Domain ‘Company’*: We observe a slightly higher recall for the domain *company* than for the domain *person*. For instance, 6 out of 10 relation extraction pipelines with relation predictor for the domain *company* achieved a Recall@10% forwarded pages of more than 50%. Contrary, we observed comparatively lower measurements for relation extractions that contain attribute values of the type person or location, such as for the relation extractor ‘*family relation*’ and ‘*person communication*’. We examined missed pages that contain relations for these types and attribute values. For instance, one problem is the recognition of relations between a lower cased person name, such as *lady gaga*, and another person that is expressed with a lower cased pronoun. In our future we will extend our feature set to capture these rare cases as well.

## B. Optimizing Relation Extraction Systems

Optimizing a relation extraction pipeline has been pioneered by authors of [3]. They propose cost models for extraction operations for an offline Web crawl scenario and for an ad-hoc Web search scenario. For instance, they propose a full scan operation that forwards each page in the corpus to a relation extractor. The index scan operator assumes an existing index of crawled pages, such as the index of a Web search engine. The index is queried for pages that likely contain instances of a particular relationship type. Each query returns a list of top-k pages that are forwarded to an extractor. As a result the relation extractor only receives pages that like contain a textual representation of a relation. In [7] we report an analysis of effective keyword query generation strategies that likely return such pages.

The filtered full scan operator and the indexed filtered scan operator utilize a relation prediction component. Authors of [3] utilize fine-tuned rules for filtering pages. Unfortunately, the study discusses two relation extractors only, such as the disease outbreak and headquarters extractor. For instance, for the disease outbreak extractor, authors of [3] report a maximum recall of 60%. Our evaluation in Section 4 is significantly more comprehensive. We compare a full scan operation with a filtered full scan operation for 18 different relation extractors. For all relation extractors we report a high recall and comparable low execution costs.

An orthogonal optimization strategy is caching extracted relations for frequently requested pages, such as Semantic Proxy [5]. The authors of System T [19] optimize the evaluation order of extraction rules. These optimizations are within the relation extractor. Contrary, our approach assumes the relation extractor as black box. Therefore these optimizations are orthogonal to our approach and may enhance the throughput of the relation extraction system further.

## VI. SUMMARY AND FUTURE WORK

Scaling relation extraction to large document collections, such as the Web, is a challenging problem. For instance, current relation extraction systems are computationally expensive: they might require several seconds to process a single page. Therefore it is too costly to sequentially scan and forward all Web pages to the extractor. However, often such an exhaustive inspection of all pages is not necessary, since only a few relevant pages contain a textual representation of a relation anyways.

We presented a relation predictor, which classifies Web pages as either relevant or irrelevant for a relation extraction task. As a classifier we trained a support vector machine that evaluates pages on a sentence level, where each sentence is transformed into a token representation of shallow text features.

In a broad experimental evaluation on more than hundred thousand pages we demonstrated that our technique robustly predicts relevant pages for 18 different relation extractors with varying attribute types. Most importantly, our relation predictor is two orders of magnitude faster than a relation

extractor. The extraction process can be speed up by at least a factor of two.

This tremendous increase in processing time per page allows us to deploy existing relation extraction systems at a large scale and for a wider range of applications than previously possible. For instance, we currently test our relation predictor technique within a highly interactive analytical Web search engine that can be reached at [www.goolap.info](http://www.goolap.info) [8]. As part of our future work we plan to release our relation predictor to the research community as a standard building block for scalable information extraction over the Web at large.

## VII. ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° FP7-ICT-2009-5-257859, 'Risk and Opportunity management of huge-scale BUSINESS community cooperation' (ROBUST).

## REFERENCES

- [1] Kasneci G., Ramanath M, Suchanek F.M., Weikum G.: The YAGO-NAGA approach to knowledge discovery. SIGMOD Row 37(4): 41-47 (2008)
- [2] Etzioni, O., Banko M., Soderland S., Weld, D.S.: Open information extraction from the Web. Commun. ACM 51(12): 68-74 (2008)
- [3] Ipeirotis, P. G., Agichtein, E., Jain, P., and Gravano, L. 2006. To search or to crawl?: towards a query optimizer for text-centric tasks. SIGMOD '06. ACM, New York, NY
- [4] YahooBoss <http://developer.yahoo.com/search/boss> (Last visited 01/10/10)
- [5] OpenCalais.<http://www.opencalais.com/documentation/calais-web-service-api/> (Last visited 01/10/10)
- [6] Feldman R., Regev Y., Gorodetsky M.: A modular information extraction system. Intell. Data Anal. 12(1): 51-71 (2008)
- [7] Löser, A., Nagel, C., Pieper, S. Augmenting Tables by Self-Supervised Web search. BIRTE Workshop at VLDB 2010
- [8] Goolap.info. [www.goolap.info](http://www.goolap.info) (Last visited 01/10/10)
- [9] Chakrabarti S., Sunita Sarawagi, Sudarshan S.: Enhancing Search with Structure. IEEE Data Eng. Bull. 33(1): 3-24 (2010)
- [10] Christensen J., Mausam, Soderland S. Etzioni O.: Semantic Role Labeling for Open Information Extraction. NAACL HLT 2010.
- [11] Kohlschütter C., Fankhauser P., Nejdil W.: Boilerplate detection using shallow text features. WSDM 2010: 441-450
- [12] Wu F., Weld. D.S.: Open Information Extraction using Wikipedia. ACL 2010
- [13] Vapnik V. The Nature of Statistical Learning Theory. New York, 1995.
- [14] Fan R.-E., Chang K.-W., Hsieh C.-J., Wang X.-R., Lin C.-J.. LIBLINEAR: A library for large linear classification Journal of Machine Learning Research 9(2008), 1871-1874.
- [15] Hoffmann, R., Zhang C. and Weld, D. "Learning 5000 Relational Extractors" ACL 2010.
- [16] Riloff, E. 1996. Automatically generating extraction patterns from untagged text. AAAI, 1996: 1044-1049
- [17] Lewis, D. D. and Tong, R. M. Text filtering in MUC-3 and MUC-4. MUC 1992
- [18] Grishman R., Huttunen S., Yangarber R. Information extraction for enhanced access to disease outbreak reports. Journal of Biomedical Informatics 35(4): 236-246 (2002)
- [19] Chiticariu L., Krishnamurthy R., Li Y., Raghavan S., Reiss F., Vaithyanathan S. SystemT: an Algebraic Approach to Declarative Information Extraction. ACL 2010
- [20] Joachims, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. ECML 1998.
- [21] Jain A, Srivastava D.: Exploring a Few Good Tuples from Text Databases. ICDE 2009: 616-62