Technische Universität Berlin

# Classification Algorithms for Web Text Filtering

**DIPLOMARBEIT**

**zur Erlangung des akademischen Grades**
**Diplom-Ingenieur**
**(Dipl.-Ing.)**

**eingereicht am**
**Fachgebiet Datenbanksysteme und Informationsmanagement**
**(DIMA)**
**Fakultät Elektrotechnik und Informatik**
**der Technischen Universität Berlin**

von

Herrn Christoph Boden (232871)

geboren am 04.07.1983 in Berlin

Gutachter:

1. Prof. Dr. Volker Markl
2. Dr. Alexander Löser

## abstract

The main subject of this thesis is to examine how classification algorithms can be adapted for the task of predicting semantic relations in Web text. Relation extraction systems capture information from natural language text on Web pages and extract semantic relations between entities. 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. Even when focused crawling techniques or automatic keyword geneartion approaches in conjuction with Web search engines are applied, many of the retreived Web still do not contain any textual representation of a relation. To enable systems to be able to extract relations within an acceptable time frame, it is thus highly desierable to be able to accurately filter irrelevant web pages.

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) that evaluates 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 more than a 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.

## Zusammenfassung

# Selbständigkeitserklärung

Teile der Ergebnise dieser Arbeit werden im Rahmen des "First International Workshop on Managing Data Throughout its Lifecycle“ (DaLi) auf der IEEE International Conference on Data Engineering (ICDE) 2011 unter dem Titel "Classification Algorithms for Relation Prediction"[1] veröffentlicht.

Die selbststaendige und eigenhädige Anfertigung versichere ich an Eides statt.

Berlin, den 24. Januar 2011

Christoph Boden

# Contents

# 1 Introduction

## 1.1 Motivation

Availability of relevant information on markets, industries and technologies is essential for decision makers in businesses as well as in government agencies. Today's CEOs are aware that the effective management and exploitation of information through IT is a key factor to business success, and essential for achieving a competitive advantage. While Management Information Systems can easily aggregate internal (accounting) data into reports suitable for decision makers, researching and compiling reports on external topics such as markets, technologies and the competitive landscape is still a tedious and thus expensive process. It requires skilled analysts and consultants even though most of the information necessary is probably publicly available in the form of text documents. Unstructured text represents the largest and fastest growing source of information available to businesses, governments and authorities. It would thus be highly desirable to leverage these sources and automatically extract and aggregate the information contained in them into a report-like representation.

The process of extracting semantic content from text is called *information extraction* (also *text analytics*. It turns the unstructured information 'buried' in texts into structured data, e.g., tuples to populate a relational database. More concretely the subtask of *Relation Extraction* comprises the identification of instances of a semantic relationship in natural language text and the extraction of relevant attribute values of this relationship. Distilling information like names, dates, or amounts from naturally occurring text is a nontrivial task. Extracting and identifying semantic relations turned out to be a computationally quite expensive task. Thus, retrieving promising documents that likely contain textual information that can be extracted, becomes a crucial part of any information extraction system. One approach to accomplish this is to query commercial Web search engines with generated keywords [2, 3] and only process the top ranked documents returned.

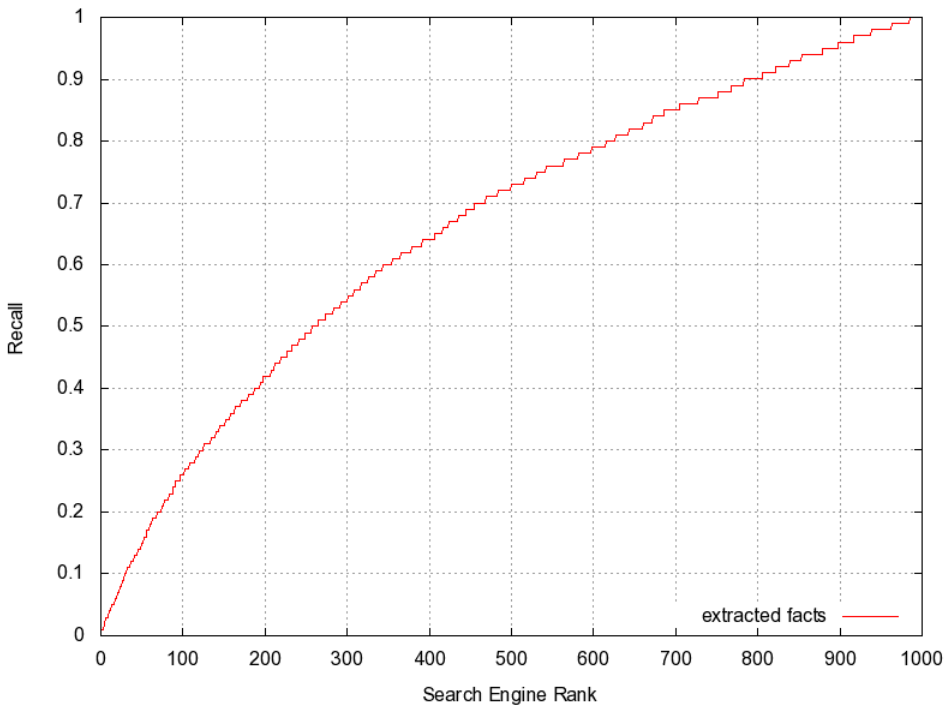


Figure 1.1: Distribution of Facts across Web Pages retrieved with a commercial Web search engine.

It turns out however that even with such a focused retrieval strategy only a tiny fraction of the documents actually contain factual information a relation extractor can identify. In a test corpus of more than 170.000 retrieved with the keyword generation strategy outlined in [3] only 2% of the pages actually contained a relation of the type in question, thus the vast majority of the retrieved documents are actually irrelevant and should not be processed by the extractor at all. It would thus be highly desirable to be able to quickly filter out these irrelevant pages to speed up the extraction process and only forward relevant pages to the relation extractor.

Figure 1.1 shows the cumulative distribution of relations across documents that have been retrieved with generated keyword queries ordered by their search engine rank. The keyword generation strategy did produce an Ordering where high ranked documents are more likely to contain a fact. However it also becomes clear that facts are found on all ranks. A heuristic that only processes the top $k$ ranked documents as proposed by [3] does omit a significant amount of facts and only lead to limited coverage. Being able to effectively filter out irrelevant pages would thus also lead to a higher recall and broader coverage of facts.

## 1.2 Problem Definition

The main subject of this thesis is to design and to implement a binary classifier as a filtering mechanism for Websites (URLs) that can accurately predict, whether a given information extractor (e.g.,*OpenCalais.org*) will be able to extract a given relationship-type with a given entity from that URL. The Websites are delivered by a web search engine, that has been queried with keywords which have been generated particularly for this relationship-type and entity. With a robust classifier one could filter the incoming URLs from the search engine and only forward promising pages.

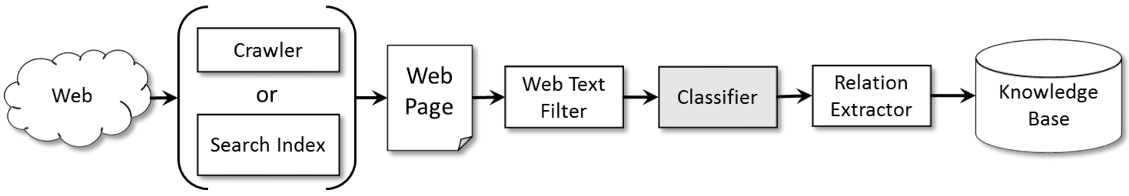


Figure 1.2: The relation extraction pipeline with the classifier

A complete relation extraction pipeline with such a classifier component is depicted in Figure 1.2. The classifier is handed the retrieved Web Pages and only forwards promising pages to the extractor. This allows scaling the system to evaluating not just top $k$, but all retrieved URLs, probably including pages containing more infrequent relations and rare information, which could ultimately contribute to a broader scope of facts extracted and thus to a more complete set of information delivered to the user.

More concretely, the following main challenges need to be solved:

- **Robust prediction**
  Thousands of possible relation extractors may process samples of the Web. These extractors may vary in their number of attributes, their attribute types or their domain specific vocabulary. The relation predictor should abstract from extractor specific words and syntactic structures and should be general applicable to these and further extractors.

- **Prediction in milliseconds**
  The relation predictor should leave the job of identifying exact attribute values, their exact border and their relation to the actual relation extractor. Ideally, the relation predictor should avoid computationally intensive natural language processing techniques, such as part-of-speech POS) tagging or deep dependency tree parsing.

As a result the relation predictor should only require tens of milliseconds for the decision to forward a relevant page to a relation extractor.

- **Balancing recall with costs**
  Since the main goal of an information extraction system is to extract as much of the structured information hidden in unstructured data as possible, any filter should not discard documents that contain factual information. Ensuring a high recall is crucial. However, very high recall comes at the price of potentially processing many irrelevant pages. Therefore the relation predictor should be able to balance between these extremes.

## 1.3 Methodology

To solve these challenges we started by examining the general structure which is used to express relationships in the English languag. Based on these observation we developed a way to generalize Web text into an abstract token representation. By transforming text into such a representation, we abstract away extractor specific words and syntactic structures. We then choose, implemented and evaluated classification algorithms based on this representation. Due to the generic abstraction, the classifiers are generally applicable to multiple extraction services, while they avoid computationally intensive natural language processing techniques, such as part-of-speech tagging or deep dependency tree parsing.

## 1.4 Thesis Outline

This thesis is organized as follows: The following chapter revisits state-of-the-art relation extraction systems and discusses related work (Chapter 2). Chapter 3 defines the task of relation prediction and presents the linguistic analysis and feature extraction approach. Chapter 4 introduces linear support vector machines as an classification algorithm and provides a brief overview of their learning theoretic foundation. Next, Chapter 5 introduces the training and evaluatin dataset, presents the evaluation metrics and discusses the results of conducted experiments. Finally, Chapter 6 summarizes the main contributions discusses future work.

# 2 Related Work

## 2.1 Relation Extraction

Relation Extraction is a language dependent and complex process. As mentioned in the introduction it can be defined as the identification of instances of a semantic relationship in natural language text and the extraction of relevant attribute values of this relationship. Various steps are executed, i.e., for extracting attribute values, determining their semantic type, their relation and the type of the detected relation.

**Web Text Filter** As a general pre-processing step, this component removes navigational elements, templates, and advertisements (so called boilerplate) from the documents [4]. It only keeps sentences and paragraphs of a page. Web text filtering is a prerequisite for relation extraction. Unfortunately, Web text filtering cannot determine if the remaining Web text contains a relation.

**Detecting Attribute Values** The goal of this step is the detection of text segments that represent potential attribute values. It typically comprises part-of-speech tagging and named entity recognition (NER). Named entity recognition is the process of identifying spans of text that constitute proper names and other special lexical structures such as temporal and numeric expressions and classifying them according to their type. Both procedures are computationally expensive. (For a general Overview of NER see [5]).

**Intra document co-reference resolution.** These techniques spot different representations of the same attribute value within or across sentences and detect missing attribute values. However, this analysis comes at the price of iterating multiple times over the global syntactic structure of the entire document [6, 7].

**Sequence of items.** Coordinating conjunctions connect any two items or more items in a series. These items can be any grammatical unit (i.e., noun phrases) except main clauses. The connected items may represent a list of attribute values or a compound object. For instance, the words Berlin, London, and New York represent a sequence of locations while the words Berlin, Germany represent a compound object. Recognizing a sequence of items requires a costly part-of-speech tagging and a domain specific statistical analysis [8].

**Extracting Relations between Attribute Values** The identification of attribute values in a sentence is a prerequisite for detecting relations between them or for determining the semantic type of a relation. Determining the semantic type of a relation is accomplished either by applying a comprehensive pattern base of regular expression patterns or sequence labelers that have been trained on labeled training data. Open information extraction approaches such as [9] do not determine the semantic type for a phrase but leave this step open to the application or the human reader.

**Generic relation extraction based on structural locality.** The 'distance' (or structural locality) of two attribute values indicates a relation in a sentence. Open Information Extraction (OIE) approaches recognize short relations in English language sentence [9, 10]. A relation is detected if two noun phrases are separated by up to five tokens and if these tokens follow a common lexico-syntactic pattern for the English language. For recognizing these patterns a part-of-speech tagger is required. Open Information Extraction approaches fail for longer sentences, such as sentences that incorporate relative or subordinate clauses or negations [10, 11]. Detecting generic relations in complex structures requires a time intensive dependency tree path analysis.

**Disambiguating the semantic type of relation.** Generic relation detection is an important prerequisite for disambiguating the semantic type of a relation. Declarative languages assign a semantic type based on the linking verb and syntactic structures [12]. The authors of [13] utilize semantic roles (such as time, location or manner) that emphasize a specific relationship type. However, recognizing semantic roles leads to more than two orders of magnitude longer processing times compared to a shallow part-of-speech analysis [11].

## 2.2 Text Filtering

Most closely related to the task of predicting a relation in Web text is the usage of information extraction systems to perform text-filtering, a subtask that has been studied in the Massage Understanding Conferences (MUC) [14]. In this task, a participating system classifies documents as either being relevant or not for a given extraction scenario. Text filtering components identify documents that contain phrases that humans utilize to express a specific event or relation. For instance, one task in MUC-4 [15] is to distinguish between terrorist events and guerilla warfare events. Text filtering components at MUC-4 pre-label sentences with named entities or part-of speech information (or expect pre-labeled documents). Next, (usually handcrafted) keywords, rule-based patterns or a classifier build on word $n$-grams are used to filter out documents that do not belong to a specific relationship type. The classification based approaches relied on a set of manually labeled training documents and all of these approaches where fine-tuned towards a specific information extraction service and relationship-type.

## 2.3 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 part-of-speech tagged pages (noun phrase, verbs, and prepositions) and outputs a set of textual patterns that can be used for discovering additional relations. A plethora of different pattern learning systems enhanced this approach (see [17] for an overview). For instance, in the PROTEUS System [18], authors used a set of linguistic seed patterns to bootstrap a pattern base by automatically learning good-quality patterns from a large, general, un-annotated corpus of documents. Most recently, systems for Open Information Extraction (OIE) [9, 19] extract short (up to five tokens) and generic relations between noun phrases in English language Web text. These generic relations are extracted with few syntactic patterns on part-of-speech labeled text. Later, the same authors [11] proposed preliminary ideas to capture distant relations with the help of Semantic Role Labeling (SRL), which requires deep dependency parsing.

The approach presented in this thesis is fundamentally different from previous text filtering and pattern generation approaches. First it does not expect tagged sentences at prediction time. For instance, it does not require tags generated through part-of-speech tagging, noun phrase chunking, named entity recognition, co-reference resolution or deep parsing. As a result, relations in a page can be predicted in a few milliseconds, instead of seconds (with part-of-speech tagging) or tens of seconds (with deep parsing) [11].

Moreover, instead of fine-tuning the feature set towards a specific relationship-type, such as in MUC-4, our encoding schema is tailored to the general problem of extracting relations. As a result, our schema is generally applicable to a wide range of relation extractors.

The design of our relation predictor does not require expert knowledge for fine-tuning or handcrafting filter rules. Rather, 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. Therefore our relation predictor is 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.

## 2.4 Optimizing Relation Extraction Systems

Optimizing a relation extraction pipeline has been pioneered by authors of [20]. 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 likely contain a textual representation of a relation.

The filtered full scan operator and the indexed filtered scan operator utilize a relation prediction component. Authors of [20] utilize fine tuned rules for filtering pages. Unfortunately, the study discusses only two relation extractors, such as the disease outbreak and headquarters extractor. For instance, for the disease outbreak extractor they report a maximum recall of only 60%. Our evaluation is significantly more comprehensive. We compare a full scan operation with a filtered full scan operation based on a classifier for 18 different relation extractors. For all relation extractors we report a high recall and comparably low execution costs.

### Querying Text Databases for efficient Information Extraction

The authors of QXtract [2] approach the issue from an Information Retrieval perspective and propose an automatic query-based technique to select documents that are promising for the extraction service. Starting with a set of user generated example tuples of the relationship type to be extracted, the system samples a set of seed documents which are forwarded to the extraction service to identify which documents are useful for the extraction task. QXtract then uses these documents to automatically generate standard search engine queries to retrieve only the relevant documents for a relation extraction task. The authors propose three approaches to generate keyword queries: as an automatic query expansion problem with the OKAPI term weighting scheme and techniques that exploit the output of two machine learning classifiers, namely decision tree based RIPPER, and Support Vector Machines (SVMs). They evaluated a combination of these three techniques approach on the two relation extractors *DiseaseOutbreak (Location, Disease, Country, Date)* and *Headquarter (organization, location)* and achieved a recall of about 20% while processing 10% of the documents in the test corpus.

In [3] the authors report an analysis of effective keyword query generation strategies based on a version of the popular discriminative category matching (DCM) scheme used in document classification [21]. This approach extracts keyword-phrases based on general syntactic patterns and ranks them by clarity and significance.

An orthogonal optimization strategy is caching extracted relations for frequently requested pages, such as *Semantic Proxy* [22]. The authors of System T [23] optimize the evaluation order of extraction rules. These optimizations are implemented within the relation extractor. Contrary, in this thesis the relation extractor is assumed to be a black box. Therefore these optimizations are orthogonal and may enhance the throughput of the relation extraction system further in addition to any effect of a classifier for relation prediction.

## 2.5 Extraction Prioritization

The authors of [24] approach the problem of having an intractable number of Web pages to process by prioritizing them. They order the documents by the expected contribution to the extraction results and process the most promising documents first.

The authors introduce a metric called Consensus Graph (CG) value, that takes into account the search demand for the information extracted form a given page. It is higher if the entities and relationships extracted satisfy more user queries. The Consensus Graph value is essentially a generalized coverage measure, as it considers the importance of individual pieces of information based on how often they are requested by the users rather than just incorporating the absolute number of extracted entities and relationships.

To rank and thus prioritize pages they estimate the utility of each page which incorporates the probability that extraction from the page will be successful and the potential contribution of the extracted facts to the consensus graph. To estimate the expected contribution of a page, the set of entities on that page is approximated by performing highly lightweight extraction on the page. This lightweight extraction searches for known entities in either the metadata and easily accessible easily accessible portions of the page content (e.g., header and footer). The estimated contribution is then computed as the delta of the current CG and the CG with the approximative set of entities on the page. Experiments show that this CG value estimation outperforms a random sample strategy.

## 2.6 Summary

Relation extraction is a computationally expensive task that contains various steps such as extracting attribute values, determining their semantic type, their relation and the type of the detected relation. So far no algorithm has been proposed that can rapidly and accurately predict whether a given relation can be extracted from a given document robustly across various relationship-types. The participants of the Message Understanding Conferences (MUC) 3 and 4 presented various approaches for the task of text filtering. However these were fine-tuned towards a specific relationship-type and require NER and POS tagged sentences. Keyword generation strategies [3, 2, 20] have been proposed to optimize the extraction process and retrieve only relevant pages that likely contain a relation. However even with those approaches, only a fraction of the retrieved documents actually contain a desired fact.

# 3 Relation Prediction

## 3.1 Problem Overview

The task of *relation prediction* is to accurately predict whether a given relation extractor (e.g., *OpenCalais.org*) will be able to extract a given relationship-type (e.g., *CompanyProduct* - the fact that a company sells a certain product) with a given entity (e.g., *VMWare*) as an attribute value. The Web pages are delivered by a Web search engine that has been queried with keywords, which have been generated particularly for this relationship-type. The relation predictor itself however, should be oblivious to the relationship-type and entity to be able to robustly predict across domains.

Note that the terminology in information extraction slightly differs from that of relational database models. To clarify the terminology, the major terms are explained in Table 3.1.

| Name | Description |
|---|---|
| Entity Type | Major category of named entities |
| | *Examples: Company, Person, Location* |
| Entity | Instance of named entities |
| | *Examples: Apple, Barack Obama, Santa Barbara* |
| Relation | A semantic relation between two or more entities |
| | *Example: based_in(Apple, Cupertino)* |
| Relationship Type | A general type of a semantic relation between two or more entity types |
| | *Example: CompanyProduct, PersonCareer, CompanyLocation* |

Table 3.1: Terminology of Relation Extraction

There are two major approaches to tackling the problem of relation prediction. As with the task of relation extraction itself one could use explicitly specified regular expression patterns to match text segments that are likely to represent a relation. While this filtering approach typically leads to high precision, it is highly susceptible to overfitting towards individual relationship-types and thus leads to low coverage. The other approach is to view the task as a classification problem and use corpus-based machine learning techniques, which require an annotated corpus of training documents. We opted for the second alternative and generated a corpus of training documents. (Details presented in chapter 4)

### 3.1.1 Relation prediction as a classification problem

The problem of predicting a discrete random variable $Y$ from another random variable $X$ is called classification (also: pattern recognition, supervised learning, discrimination). In our case the label $Y$ represents whether a document contains a certain fact or not and each document is represented by a set of feature attributes in the form of a feature vector $X$. Our annotated corpus thus consists of training examples $(x_i, y_i) \in \mathbb{R}^N \times \{\pm 1\}$ which have been generated from the joint distribution $P(X, Y)$. The objective of a classification algorithm is to learn a classification function $f : \mathbb{R}^N \rightarrow \{\pm 1\}$ that can accurately predict the labels $Y$ on previously unseen data (also assumed to be generated from $P(X, Y)$). By learning a classification function $f$ one actually aims to approximate the conditional probability $P(Y|X)$. By applying Bayes rule, the conditional probability can be rewritten as:

$$P(Y|X) = \frac{P(X|Y)\,P(Y)}{P(X,Y)} \propto P(X|Y)\,P(Y)$$

There are two major approaches to tackling this classification problem. *Generative classifiers* directly estimate parameters for $P(Y)$ and $P(X|Y)$. *Discriminative classifiers* on the other hand either try to estimate the parameters of $P(Y|X)$ directly or to learn an optimal and robust decision function that might not have a probabilistic interpretation at all, but is guaranteed to generalize well on unseen data.

The other central questions to address next to the issue of how to learn the classification function $f$, is how to extract features from an individual document to generate the feature vector $X$. The rest of this chapter introduces the feature extraction approach used in this thesis.

## 3.2 Linguistic Feature Analysis

The most straight-forward approach would be to tokenize the entire text of a document and to encode occurrences of individual tokens as features - an approach generally known as an *n-gram model*. This means that each document from a document collection $D = \{d_1,...,d_m\}$ is treated as a *bag of words* in which the order of words $w_i$ is irrelevant and only the mere occurrence of a single token (unigram) $w_i \in d_j$ or the frequency with which it occurs is used as a feature value. However such an approach would be fine-tuned towards individual relationship-types and not generalize well as each relationship-type has its own set of discriminative words. To be able to robustly predict relations across domains one has to abstract away from actual words and generalize sentences. To find a proper encoding, the structure of the way relations are expressed in natural language text has to be analyzed.

**General Model of Relationships in the English Language?** To examine the question whether there exists a general model of relationships in the English language, Etzioni et al. [25] explored a small test set of 500 sentences which had been randomly sampled from the Information Extraction training corpus developed by Bunescu and Mooney [26]. They used part-of-speech tagging to generalize the sentences and discovered that most binary relationships can be characterized by a compact set of relation-independent patterns. They identified eight major patterns that are listed in figure 3.2 to which 95% of the randomly selected sentences could be allocated. This technique is also applied by Loeser et. al [3] to extract discriminative keyword-phrases for individual relationship-types.

This is an interesting observation that very well could be exploited to predict the presence of a relation in Web text. However this type of analysis depends on part-of-speech tagging which is a somewhat expensive operation when trying to achieve high accuracy. To avoid the cost of POS-Tagging we decided to transform each sentence into a token representation that captures the information as good as possible and takes the presence of a list of extracted keywords, that express a specific relation and the knowledge of the entity in question, into account. We furthermore choose to evaluate each Web page on a sentence level rather than treating the whole page as a bag of words to honour the fact that most semantic relations are expressed within a single sentence.

**Example of the Token Representation:**

| Relative Frequency in % | Category | Simplified Lexico Syntactic Pattern |
|---|---|---|
| 37.8 | Verb | $E_1$ Verb $E_2$ |
| | | *X established Y* |
| 22.8 | Noun + Preposition | $E_1$ NP[1] Preposition $E_2$ |
| | | *X settlement with Y* |
| 16.0 | Verb + Preposition | $E_1$ Verb Preposition $E_2$ |
| | | *X moved to Y* |
| 9.4 | Infinitive | $E_1$ to Verb $E_2$ |
| | | *X plans to acquire Y* |
| 5.2 | Modifier | $E_1$ Verb $E_2$ Noun |
| | | *X is Y winner* |
| 1.8 | $Coordinate_n$ | $E_1$ (and\|,\|-\|:) $E_2$ NP |
| | | *X-Y deal* |
| 1.0 | $Coordinate_v$ | $E_1$ (and\|,) $E_2$ Verb |
| | | *X, Y merge* |
| 0.8 | Appositive | $E_1$ NP (:\|,)? $E_2$ |
| | | *X hometown: Y* |

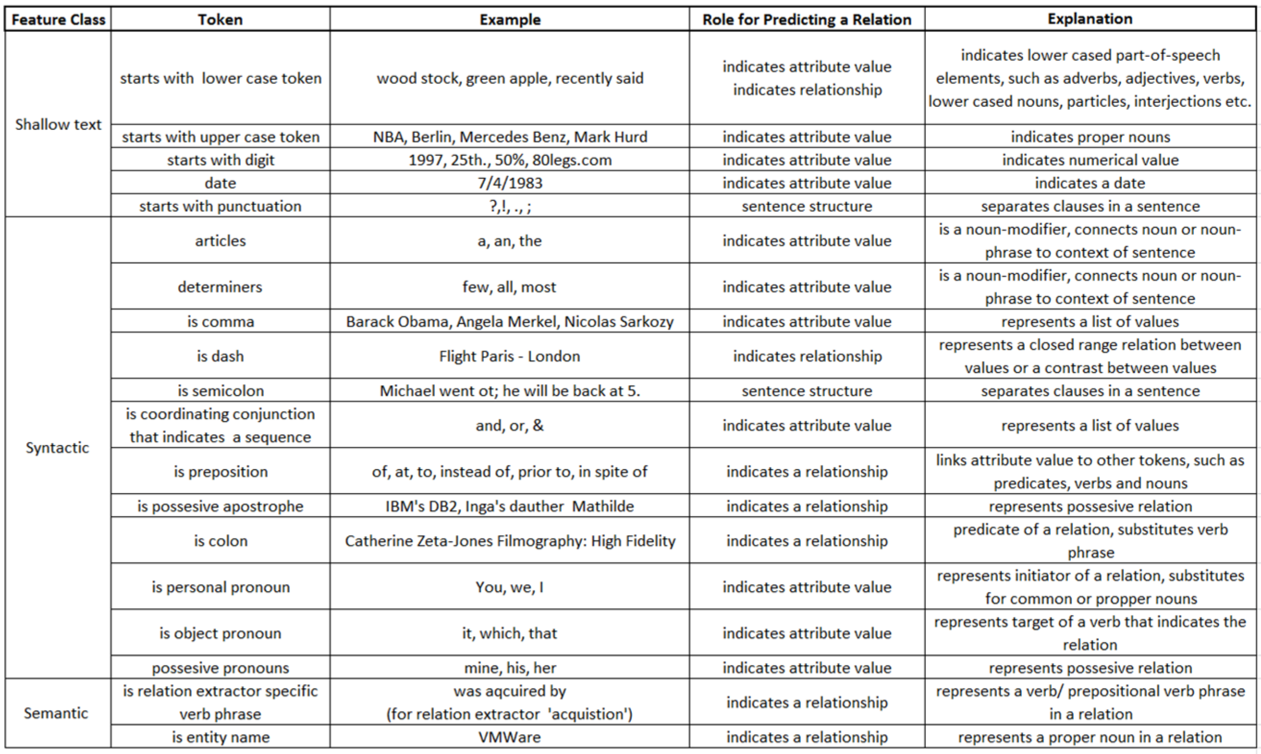
Table 3.2: Taxonomy of binary relationships

Entity: *Oracle*
Relation: *Acquisition*

```
"Oracle, building on a run of more than 65 acquisitions during the past
five years, is looking to purchase semiconductor companies and makers of
industry- specific software, Chief Executive Officer Larry Ellison said."
```

is transformed into

```
[<START>, <entity>, <comma>, <lower>, <preposition>, <article>,
<lower>, <preposition>, <determiner>, <preposition>, <number>, <lower>,
<preposition>, <article>, <preposition>, <lower>, <lower>, <comma>,
<lower>, <lower>, <keyword>, <lower>, <lower>, <conjunction>, <lower>,
<preposition>, <lower>, <lower>, <comma>, <upper>, <upper>, <upper>,
<upper>, <upper>, <lower>, <END>]
```



| Feature Class | Token | Example | Role for Predicting a Relation | Explanation |
|---|---|---|---|---|
| Shallow text | starts with lower case token | wood stock, green apple, recently said | 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 | NBA, Berlin, Mercedes Benz, Mark Hurd | indicates attribute value | indicates proper nouns |
| | starts with digit | 1997, 25th., 50%, 80legs.com | indicates attribute value | indicates numerical value |
| | date | 7/4/1983 | indicates attribute value | indicates a date |
| | starts with punctuation | ?,!, ., ; | sentence structure | separates clauses in a sentence |
| Syntactic | articles | a, an, the | indicates attribute value | is a noun-modifier, connects noun or noun-phrase to context of sentence |
| | determiners | few, all, most | indicates attribute value | is a noun-modifier, connects noun or noun-phrase to context of sentence |
| | is comma | Barack Obama, Angela Merkel, Nicolas Sarkozy | indicates attribute value | represents a list of values |
| | is dash | Flight Paris - London | indicates relationship | represents a closed range relation between values or a contrast between values |
| | is semicolon | Michael went ot; he will be back at 5. | sentence structure | separates clauses in a sentence |
| | is coordinating conjunction that indicates a sequence | and, or, & | indicates attribute value | represents a list of values |
| | is preposition | 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 |
| | is possesive apostrophe | IBM's DB2, Inga's dauther Mathilde | indicates a relationship | represents possesive relation |
| | is colon | Catherine Zeta-Jones Filmography: High Fidelity | indicates a relationship | predicate of a relation, substitutes verb phrase |
| | is personal pronoun | You, we, I | indicates attribute value | represents initiator of a relation, substitutes for common or propper nouns |
| | is object pronoun | it, which, that | indicates attribute value | represents target of a verb that indicates the relation |
| | possesive pronouns | mine, his, her | indicates attribute value | represents possesive relation |
| Semantic | is relation extractor specific verb phrase | was aqcuired by (for relation extractor 'acquistion') | indicates a relationship | represents a verb/ prepositional verb phrase in a relation |
| | is entity name | VMWare | indicates a relationship | represents a proper noun in a relation |

Figure 3.1: A list of the lexical, syntactic and semantic tags used for generalizing sentences

### Features based on shallow text and hash lists

We relax some strict requirements of a relation extractor, such as extracting exact attribute value borders, disambiguating exact semantics for attribute or relationship types and co-referencing potential attribute values within and across sentences. As a result, we significantly reduce the feature set to 19 tags only. For extracting these features we utilize shallow text features and hash lists. Figure 3.2 gives an overview our set of feature tags. In the reminder of this section we illustrate our principles for extracting lexical, syntactic and semantic features of attribute values and relations. An overview of these features is also given in Figure 3.1.

### 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 for a hash list. We encode any lowercased tokens, such as verbs and nouns, as `<lower>`, tokens starting with an uppercase character, such as proper nouns or acronyms, as `<upper>` and tokens starting with a digit as `<number>`.

### 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 to distinguish lower cased nouns from other lowercased tokens. We encode English language prepositions, coordinating conjunctions, articles and pronouns as hash list. Since we deemed them to be of particular significance, we divided the pronouns into object (*it, which, that, ...*), person (*I, you, he ...*) and other pronouns.

### Special case: Punctuation

Sequence of noun phrases, proper nouns or digits are a rich source of attribute values. We encode the comma as <comma>, colon <colon> and the semicolon as <semicolon>. This notion is based on the observation that colon frequently indicates a list of facts that can be extracted whilst the semi-colon typically indicates a sentence fragment that cannot be processed by a relation extractor

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

- **Complex noun:**
  `the wooden tool` → `<article>, <lower>, <lower>`
- **Proper noun:**
  `SAP AG` → `<upper>, <upper>`
- **Acronym:**
  `IBM` → `<upper>`
- **Complex proper noun:**
  `Sony VX 5a` → `<upper>, <upper>, <number>`

- **List of proper nouns:**
  `Oracle Inc, SAP AG and IBM` → `<upper>, <upper>, <comma>, <upper>, <upper>, <conjunction>, <upper>`
- **Date example:**
  `25 September 1969` → `<number>, <upper>, <number>`

### Verb phrases as hash list

Verb phrases and prepositional verb phrases frequently indicate a relation between two entities in English language Web text [9]. 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 `<keyword>`. 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 [12].

Sometimes we cannot access these dictionaries directly, for instance if the relation extractor is based on a machine learning approach or if we utilize a remote relation extraction service. 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 [3] and observe prepositional verb phrases in returned pages. The method is efficient and requires only a few hundred pages. Moreover, the method is effective: It omits ambiguous phrases that appear with multiple relation extractors, and only returns most significant phrases that frequently correlate with a particular relation extractor.

### Special case: Compound objects as shallow text features

Typically these phrases express a relation between an entity and an attribute of the entity. We encode phrases that represent a compound subject or object (see [9]) with shallow text features.

#### Special case: Appositive and possessive relations as shallow text features

We add a tag <_s> for encoding relations that are expressed with a possessive apostrophe. Furthermore we use the already introduced colon tag for encoding relations that are expressed as appositive phrase.

**Example:** *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:**
  *Relation Extractor: ACQUISITION (Acquirer, Acquired)*

  `SAP AG recently acquired performance management vendor Cartesis.`
  `→ <upper>, <upper>, <lower>, <keyword>, <lower>, <lower>, <lower>, <upper>`

- **Relation with prepositional verb phrase:**
  *Relation Extractor: BORN_ON_DATE(Person, Date)*

  `Catherine Zeta-Jones was born on 25 September 1969.`
  `→ <upper>, <upper>, <dash>, <upper>, <lower>, <lower>, <preposition>, <number>, <upper>, <number>`

- **Compound subject/object with noun phrase:**
  *Relation Extractor: PERSON_POSITION(Person, Position)*

  `Chancellor Angela Merkel announced → <upper>, <upper>, <upper>, <lower>`

- **Compound subject/object with prepositional noun phrase:**
  *Relation Extractor: PERSON_POSITION(Person, Position)*

  `We made an interview with Jean-Luc, Lead Engineer of Godesys AG.`

```
→ <personPronoun>, <lower>, <article>, <lower>, <preposition>,
<upper>, <dash>, <upper>, <comma>, <upper>, <upper>, <preposition>,
<upper>, <upper>
```

- **Relation with possessive apostrophe:**
  *Relation Extractor: PERSON_POSITION(Person, Position)*

  ```
  said Berlin's politically ambitious mayor, Klaus Wowereit on a press
  conference yesterday.
  → <lower>, <upper>, <_s>, <lower>, <lower>, <lower>, <comma>,
  <upper>, <upper>, <preposition>, <article>, <lower>, <lower>,
  <lower>
  ```

- **Compound subject/object with coordinating conjunction and possessive pronoun:** *Relation Extractor: TEAM_MATE (Person, Person)*

  ```
  Schuhmacher and his team mate Barrichello screamed
  → <upper>, <conjunction>, <pronoun>, <lower>, <lower>, <upper>,
  <lower>
  ```

- **Appositive relation with colon:**
  *Relation Extractor: PLAYED_IN(Actor, Movie)*

  ```
  Catherine Zeta-Jones Filmography:  High Fidelity
  → <upper>, <upper>, <dash>, <upper>, <upper>, <colon>, <upper>,
  <upper>
  ```

#### Token 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 and 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 `<lower><lower>`, `<lower><preposition>` or `<keyword>`. We select the encoding <keyword> since it belongs to the most discriminative feature class.

### 3.2.1 Feature extraction

Most semantic relations are expressed within a single sentence. To honour that fact, we evaluate each Web page on a sentence level rather than treating the whole page as a bag of words. First, we forward training pages to a relation extractor, which labels each sentence as either containing a specific relation or not. Next, we transform each sentence into a token representation of linguistic features as described above.

To capture and encode the structure inherent in this token representation *n-grams* of words $NG(w_1, \ldots, w_n; sentence_j)$ are extracted from these transformed sentences. There are several possible *n-gram* representations: occurrence(binary), frequency(e.g., $tf \times idf$) and probabilistic (language models). For example, the occurrence of n-grams of these tokens could be encoded as binary features $x$ to produce $l$ labeled data points of type $(X_i; Y_i); (1 \leq i \leq l)$, where $X_i$ is the feature vector in the $n$ dimensional feature space, and $Y_i \; in \, \{-1, +1\}$ denotes the class label.

### 3.2.2 Implementation details

To retrieve Web pages we query the web Service *YahooBoss* [27] to retrieve lists of ranked URLs. The Web pages of these URLs are then downloaded using a simple `wget()`.

Once Web Pages have been retrieved, they are sent through a Web Text Filter. This component removes navigational elements, templates, and advertisements (so called boilerplate). It only keeps sentences and paragraphs of a page [4]. We used Boilerplate's `ArticleSentencesExtractor` to extract the actual Web text from the documents and then applied the lingPipe[28] `SentenceModel` to segment the text into individual sentences. Each sentence is then generalized into its token representation by the `Transform(Sentence, RelationshipType, EntityID)`. It is comprised of closed word-lists based word-replacement and regular expression pattern matching.

## 3.3 Summary

The task of *relation prediction* to accurately predict whether a given relation extractor will be able to extract a relation essentially a classification problem. In this problem the label $Y$ has to be predicted from a set of features $X$ with a classification function $f : \mathbb{R}^N \to \{\pm 1\}$ that can accurately predict the labels $Y$ on previously unseen data. In this chapter I addressed how to extract features from an individual document to generate the feature vector $X$.

Etzioni et al. [9] observed that most binary relationships in the English language are expressed using only a rather small set of patterns. Inspired by this finding a method to generalize individual words to a set of shallow text, syntactic and semantic tags has been introduced. Each sentence of a document is transformed into a token representation of linguistic features. From this token representation, feature encodings can be extracted. The most promising approach to capture the structure is to encode the occurrence of *n-grams* of tokens as binary features $x$ to produce $l$ labeled data points of type $(X_i; Y_i)$.

# 4 Classification Algorithm: Support Vector Machine

The problem of detecting a relation in a Web page is essentially a binary text classification problem. I decided to use linear Support Vector Machines (SVMs), which have already been shown to produce superior accuracy on related tasks such as text categorization [29]. SVMs are maximum margin classifiers based on the structural risk minimization principle [30]. SVMs can be trained efficiently as binary classifiers and generalize well on unseen data.

**Soft-Margin Linear SVMs** Linear support vector machines find a hyperplane that separates the classes of training examples with maximum margin. This can be formulated as a convex optimization problem. We used the L2-loss soft margin SVM, which solves the following quadratic program on the training data:

$$\min_{w,\xi_i} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i^2$$
$$s.t. y_i (w \cdot x_i + b) \geq 1 - \xi_i \forall x_i$$
$$\xi_i \geq 0$$

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 $\xi_i$.

Most solvers, including LibSVM, solve the dual problem of this problem. To get the dual formulation one has to introduce langrange multipliers $\alpha_i \geq 0$ for the constraint and minimize the Lagrangian. The dual for the soft-margin svm is given by:

$$\max_{w,b} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_i \alpha_i$$
$$s.t. C \geq \alpha_i \geq 0, \forall x_i$$
$$\sum_i \alpha_i y_i = 0$$

## 4.1 Why SVMs work: Statistical Learning Theory

### 4.1.1 Structural risk minimization

As mentioned in the previous chapter, the classification problem is predicting a discrete random variable $Y$ from another random variable $X$, where we assume that the data is generated from an underlying distribution $P(X, Y)$. Classifiers aim to learn a classification function $f : \mathbb{R}^N \to \{\pm 1\}$ that minimizes the *expected loss* (or *risk*) according to some loss function:

$$R\left[f\right] = \int \frac{1}{2} \left|f(X) - Y\right| dP\left(X,Y\right)$$

Unfortunately this quantity cannot be evaluated since the distribution $P(X, Y)$ is generally unknown. Instead the function f is inferred from the training data und the true risk is thus approximated by the *empirical risk* (or *training error*):

$$R_{emp}\left[f\right] = \frac{1}{N} \sum_{i=1}^{N} \left|f(x_i) - y_i\right|$$

which leads to the so-called *empirical risk minimization induction principle* which goes to say that $f$ is to be chosen such that it minimizes $R_{emp}$.

Vapnik [30] showed that the inferred function $f$ approximates the true risk $R$ such that with probability $1 - \eta$ it is bounded by:

$$R\left[f\right] \leq R_{emp}\left[f\right] + \sqrt{\frac{d\left(\log \frac{2N}{d} + 1\right) - \log\left(\frac{\eta}{4}\right)}{N}}$$

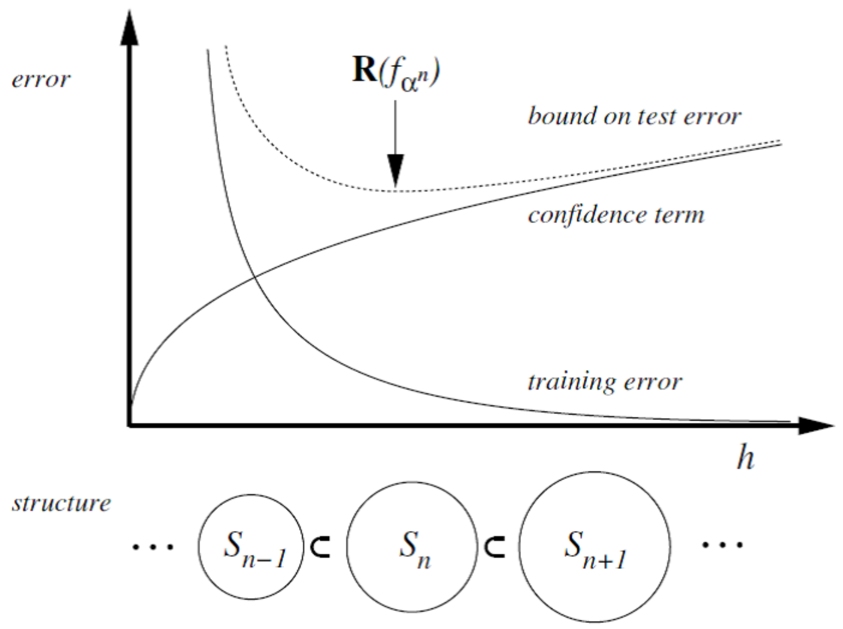


Figure 4.1: The structural risk minimization principle (SRM) illustrated. [31]

where $d$ is the so-called Vapnik-Chervonenkis (VC) dimension measuring the complexity (or *capacity*) of a function class. This leads to the already mentioned *structured risk minimization* principle which states that one has to minimize both the complexity of the function class $f^\alpha$ and the empirical risk. This principle is illustrated by Figure 4.1. The function class is decomposed into a nested sequence of subsets of increasing complexity. The SRM chooses a function $f_\alpha^n$ such that the bound on the test error is minimized. This is achieved by choosing $f$ such that it comes from an element of the structure that has a low VC-dimension (denoted by $h$ in the figure) and a low training error at the same time.

### 4.1.2 Linear hyperplane classifier

Figure 4.2 illustrates a linear hyperplane classifier with the classification function $f(x) = sgn((w \cdot x) + b)$. The optimal hyperplane separating the data points is shown as a dotted line. A hyperplane is said to be in canonical form if it has been scaled such that $\min_{x\ i} \in X\ |(w \cdot x_i) + b| = 1$ holds. The *margin* (the distance between two of the so-called support vectors that are lying on the margin) is then given by $(\frac{w}{\|w\|} \cdot (x_1 - x_2)) = \frac{2}{\|w\|}$ where $x_1$ and $x_2$ are support vectors of both classes. Finding the maximum margin hyperplane that separates the data can thus be characterized by the following optimization problem:

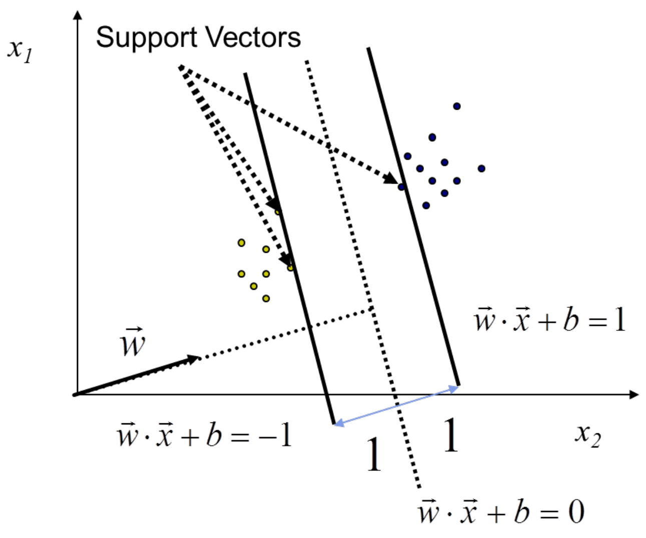

$x_1$
Support Vectors
$\vec{w}$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}\cdot\vec{x}+b=-1$
1
1
$x_2$
$\vec{w}\cdot\vec{x}+b=0$

Figure 4.2: A linear hyperplane classifier. The optimal separating hyperplane is shown as a dotted line. [32]

$$\max_{w} \frac{w^2}{\|w\|}$$

$$s.t.(w \cdot x + b) \geq 1 y_i \forall x_i \in class_1$$

$$(w \cdot x + b) \leq -1 y_i \forall x_i \in class_2$$

## 4.1.3 Maximum margin classifiers generalize well

Vapnik [?] showed that the VC-dimension $d$ for a canonical hyperplane $(w$,$b)$ in the form of $y = sgn\,(w \cdot x + b)$ scaled such that $\min_{x\,i} \in X\, |(w \cdot x_i) + b| = 1$ satisfies the following equation:

$$d \leq \min\left\{(R^2 \|w\|^2) + 1, n + 1\right\}$$

where $R$ is the radius of the smallest ball containing the data (training and test data) and $n$ is the dimensionality of the data. When this result is plugged back into the equation for the srm-bound:

$$R\left[f\right] \leq R_{emp}\left[f\right] + \sqrt{\frac{d\left(\log\frac{2N}{d}+1\right) - \log\left(\frac{\eta}{4}\right)}{N}}$$

it becomes apparent, that minimizing $\|w\|$ actually minimizes the generalization error. Maximizing the margin ($\max_w \frac{w^2}{\|w\|}$) is equivalent to minimizing the norm of the weight vector $\|w\|^2$. The objective function of the optimization problem can thus be expressed as $min\frac{1}{2}\|w\|^2$ which gives the canonical SVM-Problem introduced above. It is thus guaranteed to minimize the generalization error. If the probability distribution of the data $P(X,Y)$ is unknown, discriminative maximum margin classifiers such as the support vector machine are thus the best approach to estimate a classification algorithm that generalizes well on unseen data. The generalization error is typically estimated by evaluating a trained SVM on unseen test data.

## 4.2 Prediction with a trained SVM

At prediction time pages are first segmented into sentences. Each sentence is then transformed into its a token representation of shallow text, syntactic and semantic features as described above. Next, the occurrence of *n-grams* of these tokens is encoded as binary features $x$ to produce $l$ labeled data points 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.

Next, the label for each sentence in the document is predicted. The support vector machine classifies data points according to the learned classification function:

$$f(x) = sgn((w \cdot x) + b)$$

Note, that since SVM-Solvers usually solve the dual problem to the quadratic program presented above, the decision function is usually calculated as:

$$f(x) = sgn(\sum_{i=1}^{n} y_i \alpha_i (x, x_i) + b)$$

However this only classifies sentences. To generate a prediction for a document we simply take the maximum of all predicted labels of all sentences $s$ of a document $d$.

$$f(d) = max_{s \in d} f(s)$$

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. Note that this implicitly assumes that the sentences in a document are independent.

## 4.3 Implementation details

To solve the Support Vector Machine optimization problem I adapted the *liblinear* solver with the `L2R_L2LOSS_SVC_DUAL` solver type. With this solver there are two hyperparameters to tune: the weight of the slack $C$ and the stopping criterion $\epsilon$. These parameters are determined using a grid-search across a coarse parameter space.

The optimal set of parameters according to the Area under the Curve (AUC) metric is then used to train and evaluate the actual classifier. The details of the data set, experimental setup as well as the AUC metric will be described in Chapter 5.

## 4.4 Summary

To solve the binary text classification problem of relation prediction, linear Support Vector Machines (SVMs) are used. This discriminative Classifier can be formulated as a convex optimization problem. SVMs are maximum margin classifiers that find a hyperplane that optimally separates the classes of training examples with maximum margin. SVMs are based on the structural risk minimization principle [30] and are thus guaranteed to minimize generalization error according to statistical learning theory. The Liblinear *LibLinear* solver with the `L2R_L2LOSS_SVC_DUAL` solver type is used to train and evaluate the SVMs. Hyperparameter are tuned using grid search based on the AUC metric. At prediction time, the label for a document is generated by taking the max of all predicted labels of all sentences $s$ of a document $d$. Sentence labels are predicted using the classification function $f$ learned by the SVM.

# 5 Training and Evaluation

## 5.1 Data Set: A Corpus of tagged Web pages

To be able to consistently train and evaluate classification algorithms, a fixed corpus of annotated web Pages has been generated. We choose 18 different relationship-types of the domains Company and Person from the public extraction service OpenCalais [22]. A detailed overview of the different relation extractors for these relationship types and the schema associated with them is presented in table 5.1. The extractors significantly vary in their attribute types, in their extraction domain as well as in their number of attributes.

For each relationship type we generated relation specific keyword queries with the keyword generation strategy proposed in [3]. For each type we queried a Web search engine [27] retrieved the top-1000 results and downloaded the pages from the Web via wget. Some pages could not be downloaded however, so overall we only retrieved about 153.000 pages. Each page was also forwarded to a the relation extraction service OpenCalais [22] to annotate each page with the relations it contains.

Figure 5.1 shows the distribution of relations for different relationship-types in the downloaded corpus. Note, that because of the generated keywords, this sample is already biased towards pages that likely contain a relation. Nevertheless we observe a rather sparse distribution of relations in our corpus. For instance, relations of the type *PersonCareer* are a relatively frequent type and appear in about 6% of the pages in our corpus. Contrary, relations of the type conviction appear in less than 1% of the pages. Apparently, extraction systems still retrieve many irrelevant pages that do not contain any text that represents a relation, even though a keyword generation strategy has been used.

| Relationship-type | Schema |
|---|---|
| Acquisition | *(Company_Acquirer, Company_BeingAcquired, Date, Status)* |
| CompanyAffiliates | *(Company_Affiliate, Company_Parent, AffiliateRelationType)* |
| CompanyEmployeesNumber | *(Company, EmployeesNumber, Unit, Location, LocationType)* |
| CompanyExpansion | *(Company, ExpansionType, Location, Status, Date, DateString)* |
| CompanyFounded | *(Company, Year)* |
| CompanyLocation | *(Company, City, ProvinceOrState, CompanyLocationType)* |
| CompanyProduct | *(Company, Product, ProductType)* |
| CompanyReorganization | *(Company, Status, DateString, Date)* |
| CompanyTechnology | *(Company, Technology)* |
| CompanyTicker | *(Company, Ticker, StockExchange)* |
| Conviction | *(Person, Charge, OtherCharges, DateString, Date)* |
| EmploymentRelation | *(Person_Employer, Person_Employee, Position, Date, DateString)* |
| FamilyRelation | *(Person, Person_Relative, FamilyRelationType)* |
| PersonAttributes | *(Person, Age, BirthDate, BirthPlace, Gender)* |
| PersonCareer | *(Person, Position, Company, Organization, Country, ProvinceOrState, City, CareerType, Status)* |
| PersonCommunication | *(Person, PersonDescription, Person2, PersonDescription2, Facility, OrganizationOrCompany, Date, Status)* |
| PersonEducation | *(Person, Certification, SchoolOrOrganization)* |
| PersonTravel | *(Person, LocationOrigin, LocationDestination, DateString, Date, Status)* |

Table 5.1: The different types of relationships chosen for the experiments. Relation extractors significantly vary in their attribute types, in their extraction domain and in their number of attributes.

## 5.2 Experimental Setup

To be able to train and test a classifier each Web page has to be transformed into its feature representation $(x_1,...,x_n; y)$. Figure 5.2 illustrates the process pipeline: a web page is sent to the Relation Extractor to extract any relations contained in it. (This actually happened when the corpus was assembled.) Next, the Web page is sent through the feature extractor that first extracts the textual content of a page and then transforms the Web text into its token representation as described in chapter 3. Next the occurrence of n-grams $NG(w_1, ..., w_n; sentence_j)$ is encoded in a feature vector.

As a next step in each experiment, hyperparameters are tuned via 10-fold cross-validation. After the optimal parameter set has been found, the actual training and evaluation begins. The training data is randomly divided into 10 equi-size blocks of data. For each of the Data blocks the classification algorithm is trained on the remaining 9 blocks of data. After

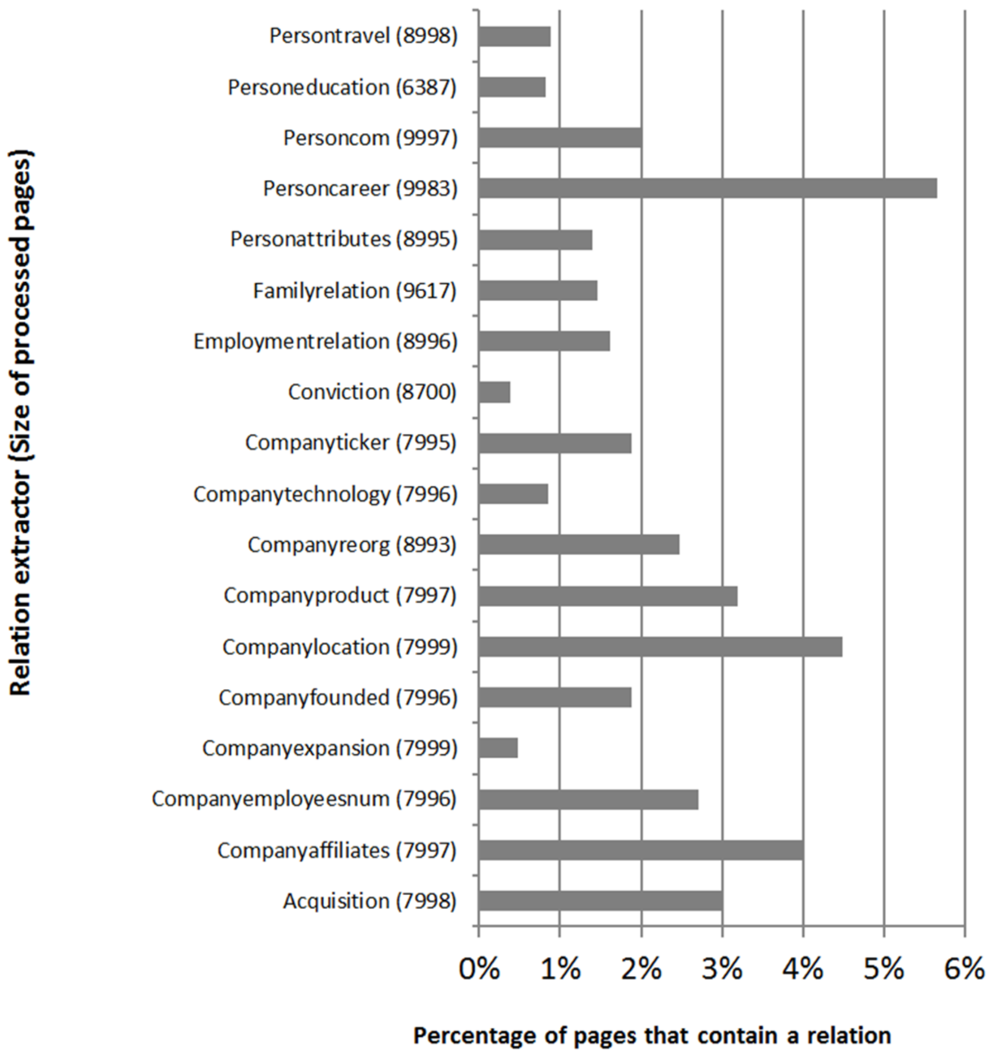


Figure 5.1: Distribution of extractable facts in the corpus of retrieved Web pages

training (solving the optimization problem) has finished, the labels for the held-out block of data are predicted.

This way the performance of the classifier is estimated by predicting labels on unseen test data. This prediction process is illustrated in figure 5.3. At prediction time each page is segmented page is transformed into its feature vector representation. The classifier (relation predictor) then predicts whether the page contains a relation or not. If the prediction is positive, the entire page is forwarded to the relation extractor. Otherwise, the page is discarded and removed from the relation extraction pipeline.

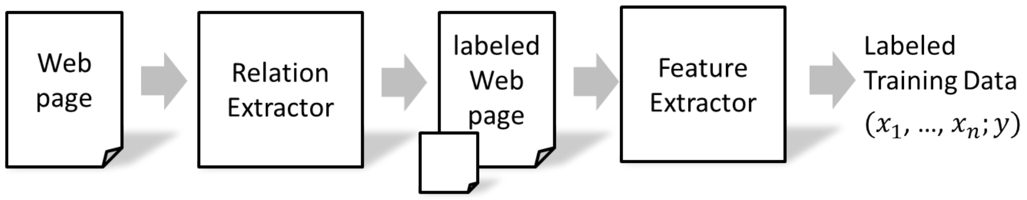


Figure 5.2: The retrieved Web pages forwarded to the Relation Extractor (*OpenCalais*) to obtain the relations embedded in the Web-Text of a page. The Web-Text is then processed by the Feature extractor described in Chapter 3 to generate the feature representation.

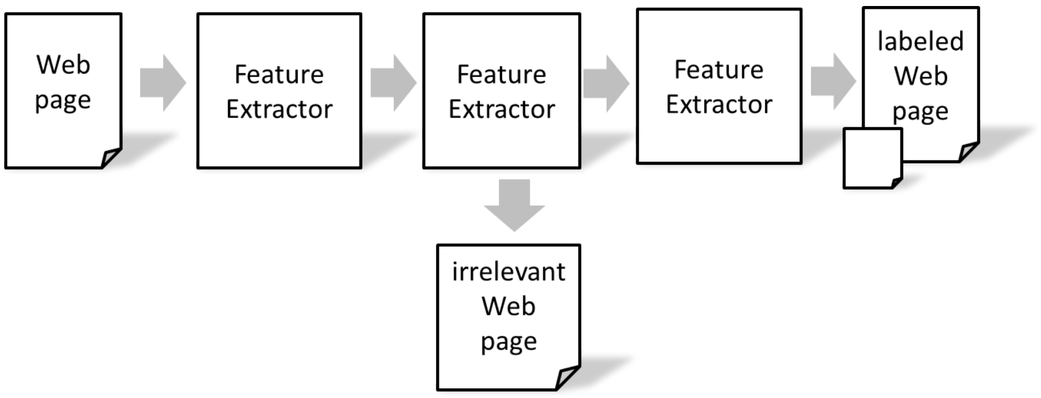
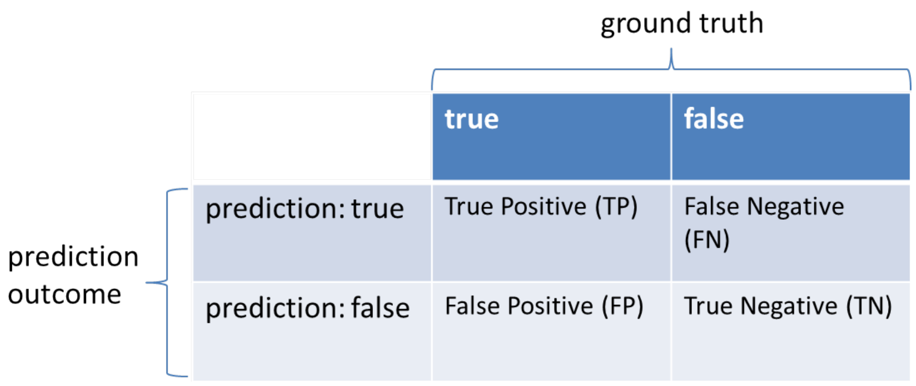


Figure 5.3: The processing pipeline of an extraction system with a classifier as a relation predictor at prediction time

## 5.3 Evaluation metrics



ground truth

prediction outcome

| | true | false |
|---|---|---|
| prediction: true | True Positive (TP) | False Negative (FN) |
| prediction: false | False Positive (FP) | True Negative (TN) |

Figure 5.4: A contingency table: illustrates the possible outcomes of predicting the label for a data point.

A binary classifier is designed to predict a label $y \in \{true, false\}$. The labels on the evaluation data represent a so called *ground truth*. To evaluate the quality of a classifier one uses the algorithm to predict the labels on this unseen data. For each data point there are four possible outcomes:

- **True Positive** The data point was predicted to be positive and actually is a member of the positive class.
- **False Positive** The data point was predicted to be positive but actually is a member of the negative class.
- **True Negative** The data point was predicted to be negative and actually is a member of the negative class.
- **False Negative** The data point was predicted to be negative but actually is a member of the positive class.

Figure 5.4 illustrates these outcomes in a schema that is generally known as a *contingency table* or *confusion matrix*. A contingency table can be seen as a snapshot of the performance of a classifier at one particular threshold level. It can also be summarized by calculating the accuracy as a standard measure to capture the quality of a classifier:

$$Accuracy = \frac{TP + TN}{|Samples|}$$

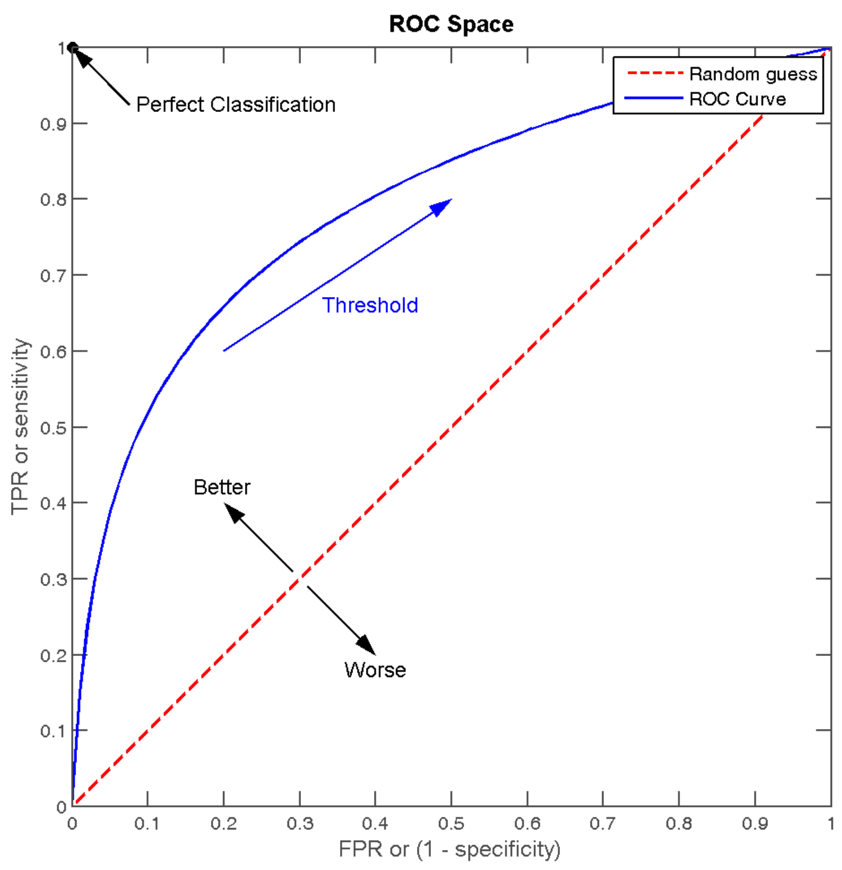


Figure 5.5: A Receiver Operating Characteristic: Illustrates the trade-off between False Positives and True Positives of a classifier as the threshold is relaxed. The area under that curve (AUC) is used as a measure of classifier performance. [33]

### 5.3.1 Receiver Operating Characteristic

To illustrate the performance of a classifier at different threshold levels, the tool of choice is called *Receiver Operating Characteristic*. Its main features are illustrated in figure 5.5. It shows the inherent trade-off between the false-positive rate and the false-negative rate of a classifier. By relaxing the threshold level for the decision function one can 'travel' along the ROC-Curve.

As already mentioned in the introduction to this chapter, the data in the corpus is unequally distributed. Only a tiny fraction of the web Pages actually contains a relation and are thus members of the positive class. The vast majority of the downloaded Web pages does not contain any relations and is thus a member of the negative class. The standard measure of *accuracy* is not an appropriate metric to evaluate classifiers on this data.

For example: simply classifying all pages as 'negative' would lead to an accuracy of about 98%. The *Area under the Curve (AUC)* of the Receiver Operating Characteristic (ROC) is a much more robust metric for classification performance and is commonly used to evaluate machine learning algorithms on unbalanced data [34][35]. The AUC represents the probability that a randomly chosen positive data sample will be assigned a higher value by the decision function of the classifier than a randomly chosen negative sample. I thus used the AUC as a performance metric to tune the Hyperparamters $C$ and $\epsilon$.

## 5.3.2 Recall-curves

While ROC-Curves are a standard way to present the behaviour of classifiers, they do not capture the actual task in question very well. The original motivation for the relation predictor was to be able to extract instances of relationships from web pages as fast as possible while processing as few Web pages as necessary. With the tagged corpus of Web pages we actually have a *Gold Standard.* It reflects the maximum number of relations that the evaluated relation extraction service could extract from the corpus.
A more intuitive way to present the performance of the classifier is to translate the true and false positives into processed pages and to measure how many pages have to be processed over all to be able to recall a certain number of relations from the Gold standard. Let *recall* be the fraction of relations in forwarded pages divided by the total number relations in the collection:

$$recall = \frac{|relations_{forwarded\_pages} \cap relations_{Gold\_Standard}|}{|relations_{Gold\_Standard}|}$$

Next to the ROC-curves already discussed I also present Recall-Curves that illustrate how many processed pages overall one has to accept when aiming for a particular recall level. In addition to the recall curve generated by the predictor the plots also contain two strategies for comparison:

- **Full Scan:** Shows the results of a full tables scan over all pages, where the relation extractor processes the pages in a random order.

- **Index Scan:** Shows the results of an index scan over all pages, where the relation extractor processes the pages in the order of the Search Engine rank.

### 5.3.3 Execution time

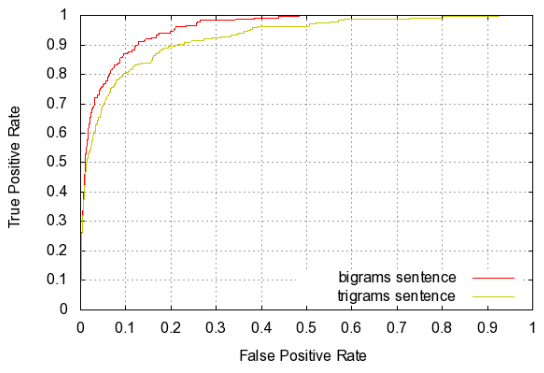
Finally I also measured the execution time of the classifiers and compared the execution time to a relation extractor. As described above an extraction service has been used to extract relations. Therefore I could only measure the total time of the extraction process, which is sending a page to the service, execution the relation extraction at the service and retrieving extracted information. Note, that authors of [11] observe similar execution times without the overhead of sending pages through the network. The prediction time includes boilerplate removal, sentence segmentation, feature extraction and the actual prediction computation.

## 5.4 Experimental Results

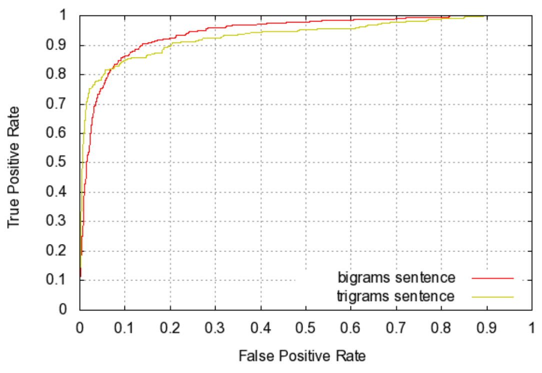
As discussed in chapter 3, most semantic relations are expressed within a single sentence. To honour that fact, Web pages are evaluated on a sentence level rather than treating the whole page as a bag of words. I conducted experiments with both bigrams and trigrams on a sentence level for each of the 18 different relationship types in the corpus. The resulting AUC Values and execution times of the experiments are presented in Table 5.2. Figures 5.6 through 5.8 show the ROC-Curves and Figures 5.4 through 5.11 the Recall curves for each relationship type.

| Relationship Type | extraction time in [ms] | BiGrams AUC | BiGrams prediction time in [ms] | TriGrams AUC | TriGrams prediction time in [ms] |
|---|---|---|---|---|---|
| Acquisition | 2288.1 | **0.958** | 22.9 | 0.926 | 23.2 |
| CompanyAffiliates | 2195.6 | **0.944** | 21.6 | 0.932 | 22.1 |
| CompanyEmployeesNumber | 2201.1 | 0.935 | 25.5 | **0.949** | 26.0 |
| CompanyExpansion | 2036.8 | **0.931** | 19.7 | 0.883 | 20.1 |
| CompanyFounded | 1951.5 | **0.933** | 19.5 | 0.931 | 19.9 |
| CompanyLocation | 1896.8 | **0.958** | 17.3 | 0.952 | 17.6 |
| CompanyProduct | 2335.1 | **0.939** | 26.5 | 0.934 | 27.0 |
| CompanyReorganization | 2350.8 | **0.961** | 34.4 | 0.954 | 33.0 |
| CompanyTechnology | 2062.6 | 0.883 | 23.9 | **0.899** | 24.6 |
| CompanyTicker | 2200.9 | **0.999** | 22.3 | 0.993 | 22.6 |
| Conviction | 3107.9 | 0.731 | 41.8 | **0.787** | 42.6 |
| EmploymentRelation | 2827.9 | **0.841** | 35.9 | 0.783 | 35.8 |
| FamilyRelation | 2886.4 | 0.772 | 41.0 | **0.799** | 40.3 |
| PersonAttributes | 2858.1 | **0.820** | 36.1 | 0.803 | 36.3 |
| PersonCareer | 3508.4 | **0.845** | 57.1 | 0.842 | 58.2 |
| PersonCommunication | 2959.1 | 0.772 | 59.5 | **0.841** | 43.7 |
| PersonEducation | 3403.0 | 0.744 | 45.1 | **0.762** | 45.3 |
| PersonTravel | 3252.6 | **0.820** | 49.4 | 0.755 | 49.5 |

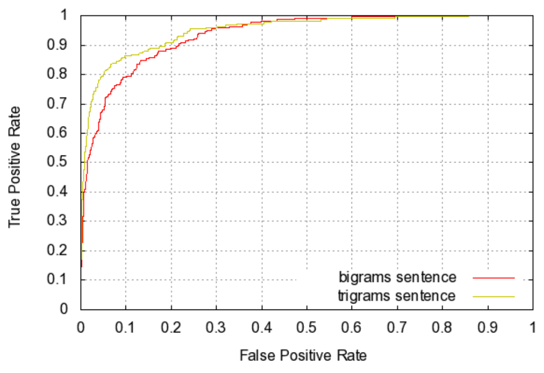
Table 5.2: This table sums up the results of the experiments. It shows the out-of-sample Area under the ROC Curve (AUC) measurements for both the bigram and trigram based sentence-level SVM Classifier. It also lists the average prediction time per processed page for the two n-gram approaches as well as the time needed per page by the actual relation extractor.



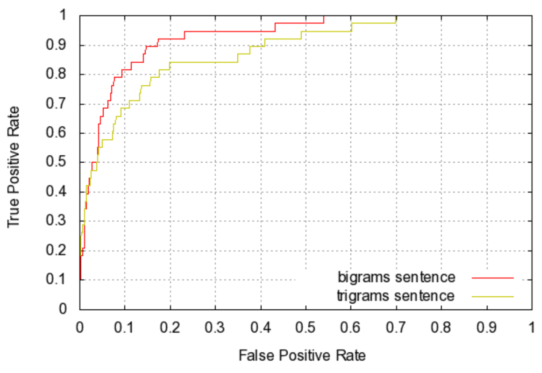
(a) ROC Curves for acquisition



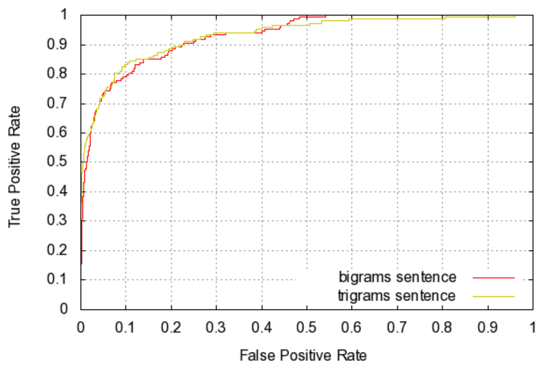
(b) ROC Curves for companyaffiliates



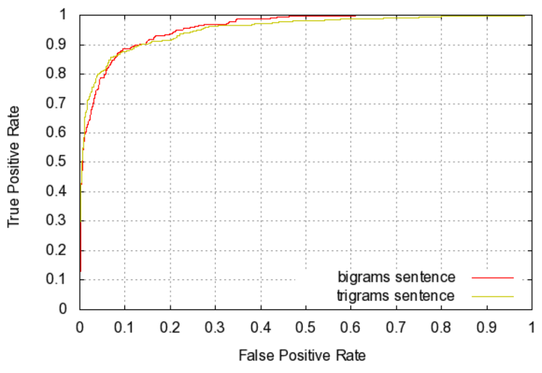
(c) ROC Curves for companyemployeesnumber



(d) ROC Curves for companyexpansion

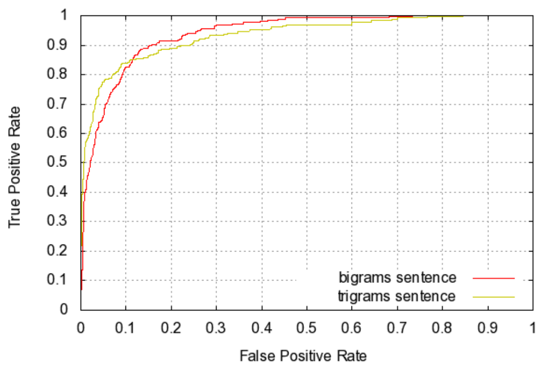


(e) ROC Curves for companyfounded

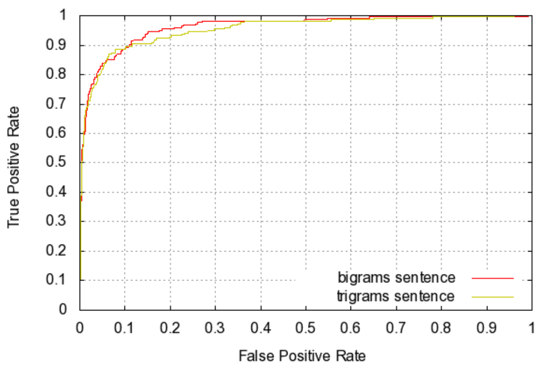


(f) ROC Curves for companylocation

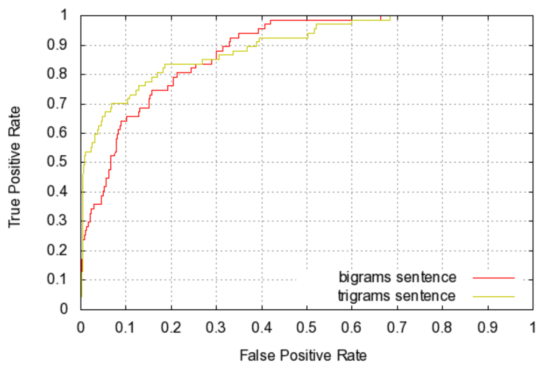
Figure 5.6: The ROC Curves for all experiments for each individual relationship-type



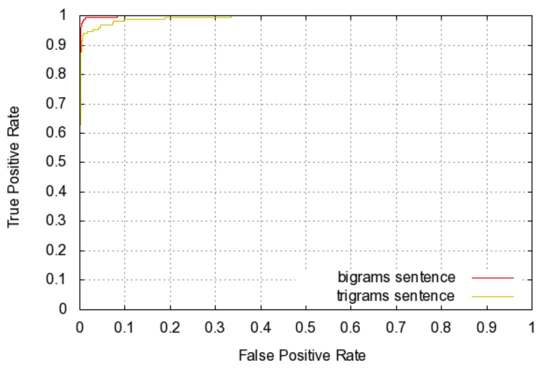
(a) ROC Curves for companyproduct



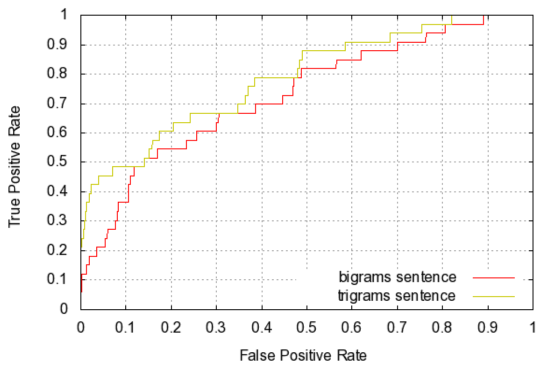
(b) ROC Curves for companyreorganization



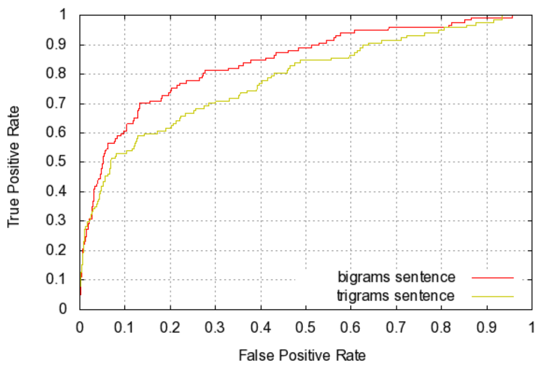
(c) ROC Curves for companytechnology



(d) ROC Curves for companyticker

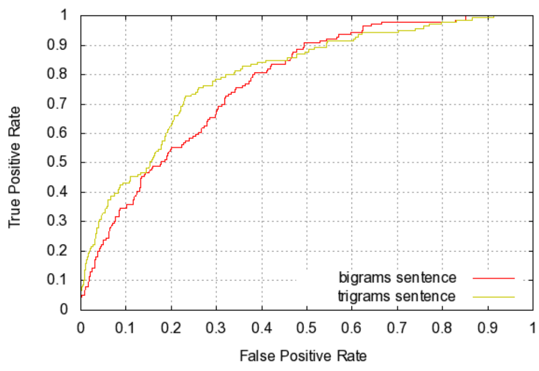


(e) ROC Curves for conviction

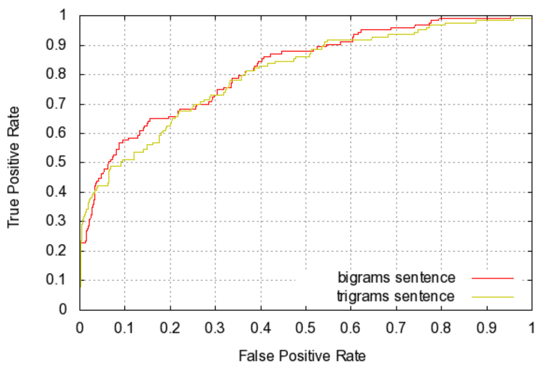


(f) ROC Curves for employmentrelation

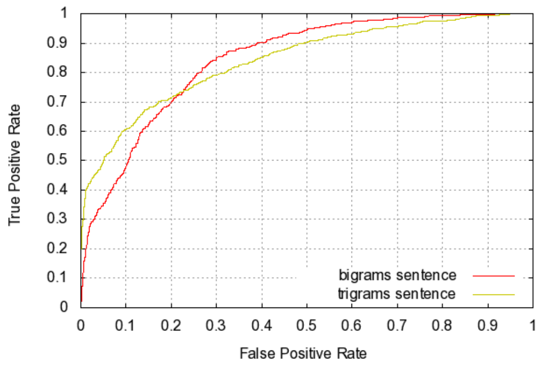
Figure 5.7: The ROC Curves for all experiments for each individual relationship-type



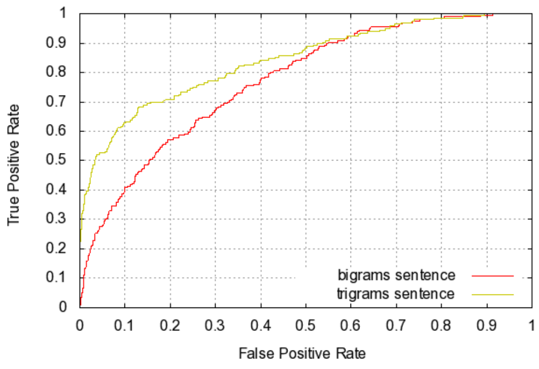
(a) ROC Curves for familyrelation



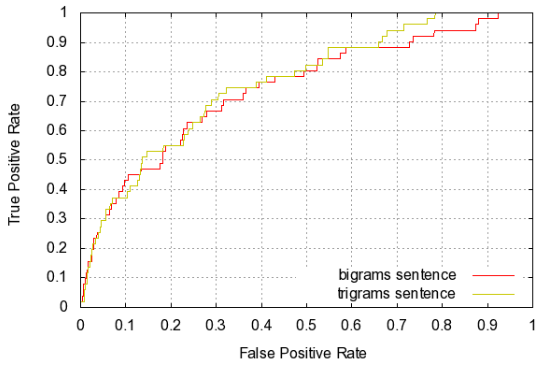
(b) ROC Curves for personattributes



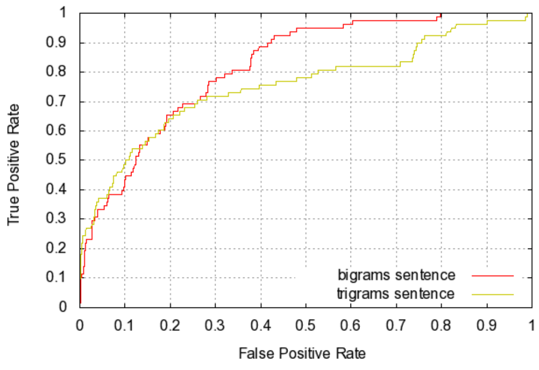
(c) ROC Curves for personcareer



(d) ROC Curves for personcommunication

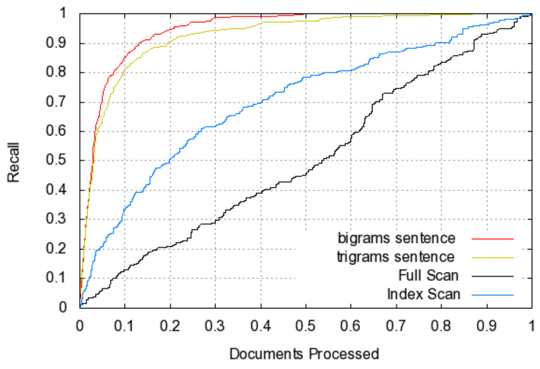


(e) ROC Curves for personeducation

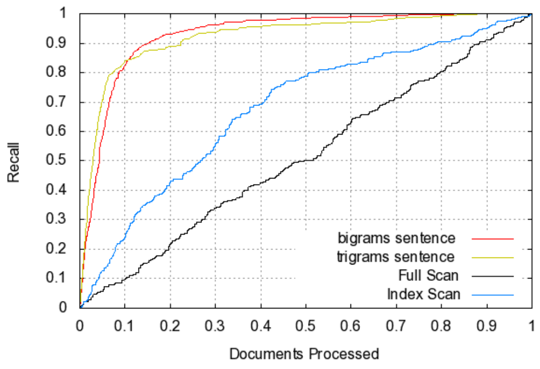


(f) ROC Curves for persontravel

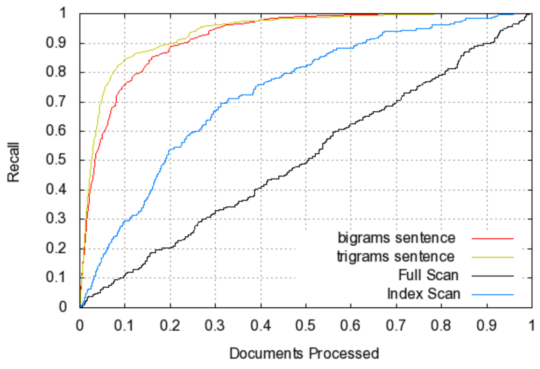
Figure 5.8: The ROC Curves for all experiments for each individual relationship-type



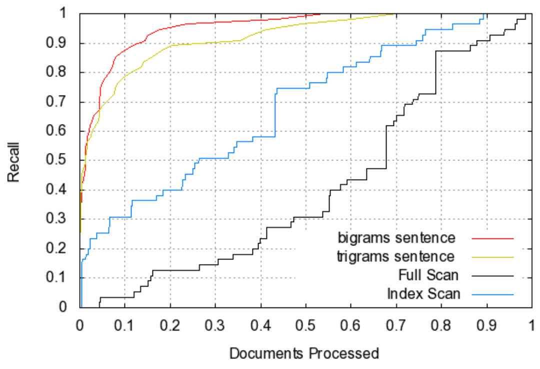
(a) Recall curves for acquisition



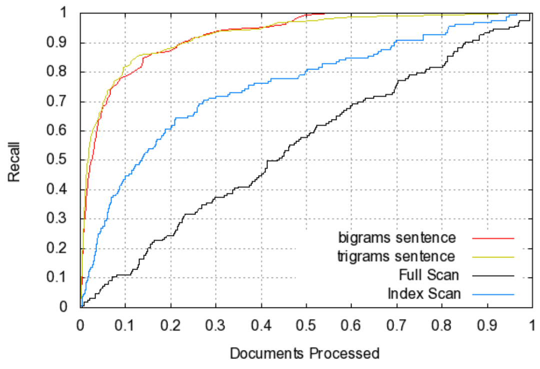
(b) Recall curves for companyaffiliates



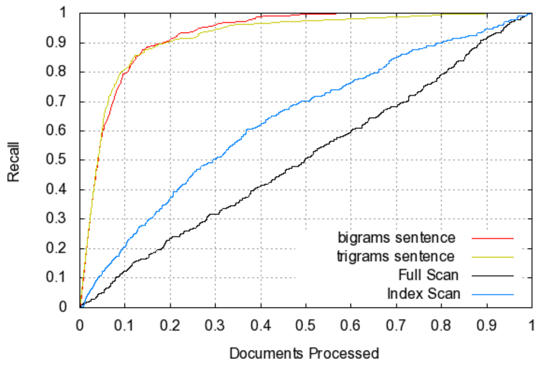
(c) Recall curves for companyemployeesnumber



(d) Recall curves for companyexpansion

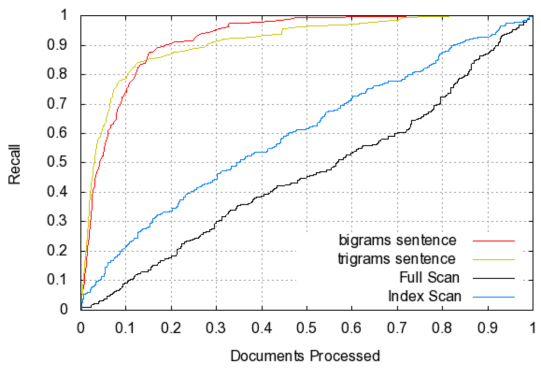


(e) Recall curves for companyfounded

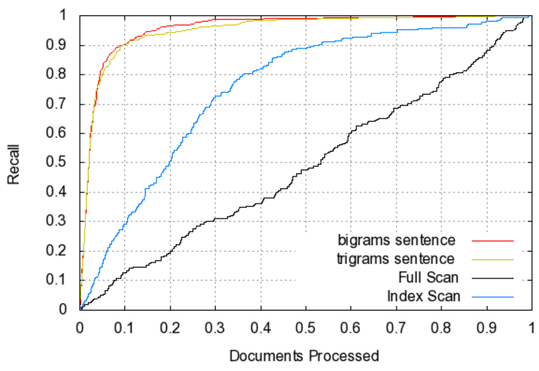


(f) Recall curves for companylocation

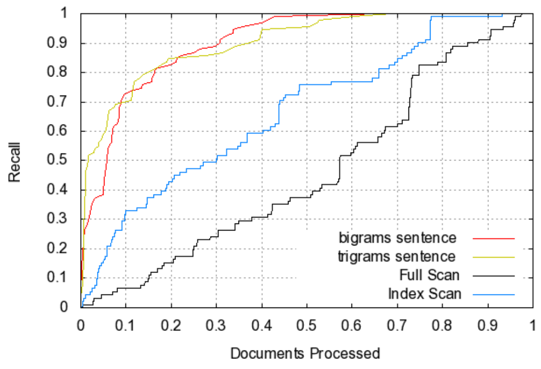
Figure 5.9: The Recall Curves for all experiments for each individual relationship-type



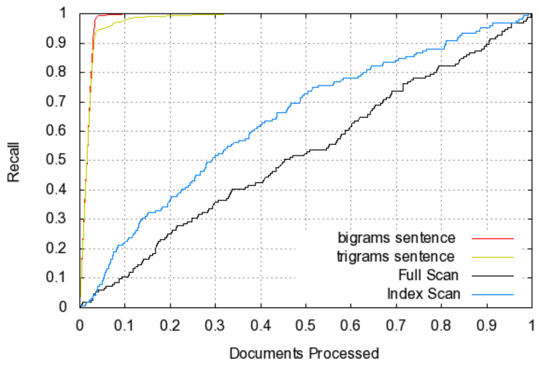
(a) Recall curves for companyproduct



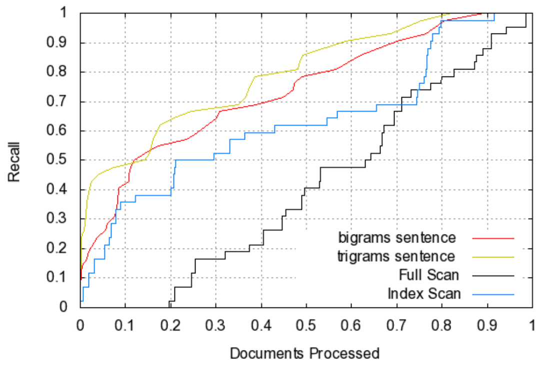
(b) Recall curves for companyreorganization



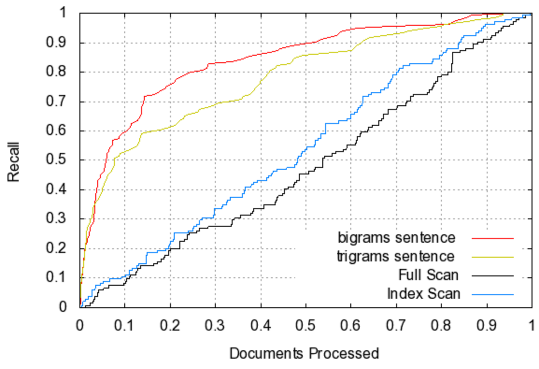
(c) Recall curves for companytechnology



(d) Recall curves for companyticker

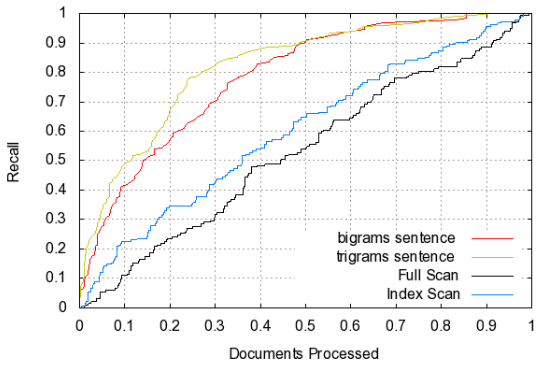


(e) Recall curves for conviction

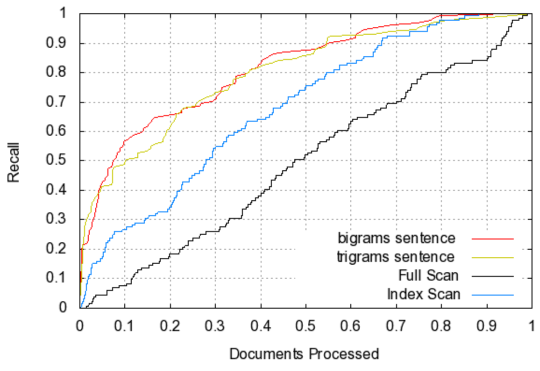


(f) Recall curves for employmentrelation

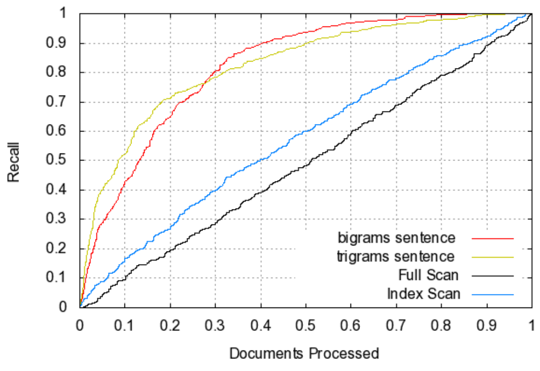
Figure 5.10: The Recall Curves for all experiments for each individual relationship-type



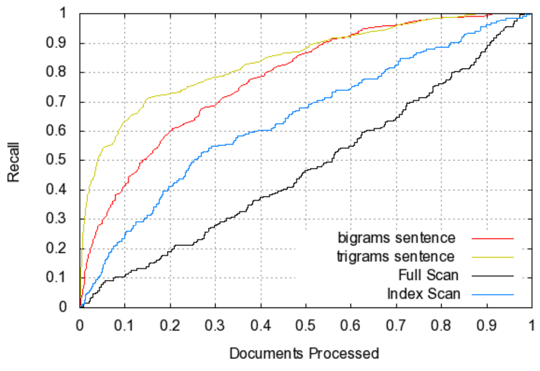
(a) Recall curves for familyrelation



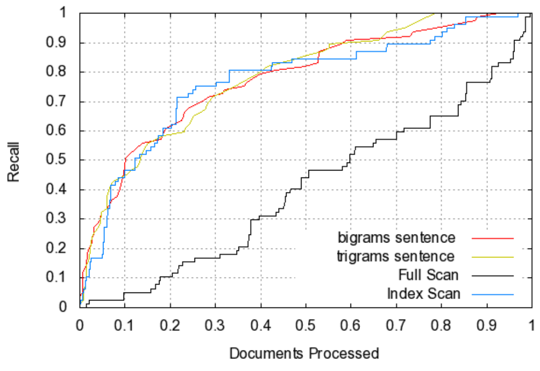
(b) Recall curves for personattributes



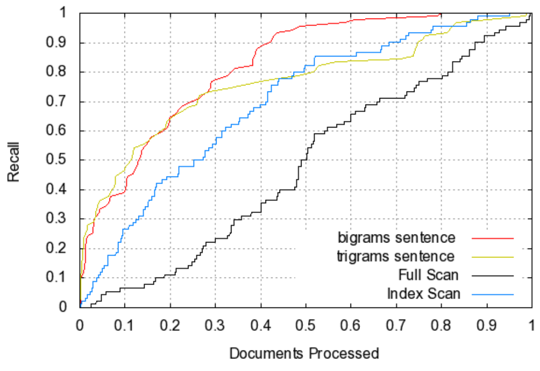
(c) Recall curves for personcareer



(d) Recall curves for personcommunication



(e) Recall curves for personeducation



(f) Recall curves for persontravel

Figure 5.11: The Recall Curves for all experiments for each individual relationship-type

### 5.4.1 Prediction in Tens of Milliseconds

Table 5.2 lists the average execution times of both the classifiers and the relation extraction service. The prediction time includes boilerplate removal, sentence segmentation, feature extraction and the actual prediction computation. The classifiers are about two orders of magnitude faster than the actual relation extraction service for both the bigram and the trigram model. This is a remarkable gain in performance. For example the predictor takes about 18 ms per Web page on the relation 'CompanyLocation' and 58 ms per Web page on the relation 'PersonCareer'.

The difference in prediction times for the bigram and trigram models is negligible. Variations in the prediction and extraction times between different relationship-types are most likely due to different average length of the text input for each relationship type. If one normalizes by the average size of the Web text inputs, the times are almost constant across relationship-types.

The classifiers are fast and maintain their performance across the examined domains. It is reasonable to suspect that this result holds across further domains as well.

### 5.4.2 Configurations for different application scenarios

Different cost budgets may appear in application scenarios. As can be seen in the recall curves shown in Figures 5.4 through 5.11, there is an inherent trade-off between the recall of facts in the corpus and the number of documents that have to be processed to achieve that recall. To gain some deeper insights into the meaning of these recall levels, it is helpful do identify common application scenarios for relation prediction by their cost budget. Let **Recall@Costs** be a measure of he recall, denoted as the fraction of retrieved relations, conditioned on the 'invested' processing costs, measured by the fraction of processed pages.

- **Recall@5% (Ad-hoc query scenario).** This scenario reflects tight budget costs, since the classifier forwards less than 10% of pages to the relation extractor. It is suitable for an ad-hoc analysis, such as proposed in [20] [36]. It may also be applied to an aggregation scenario in [3] where only few top ranked pages from a Web search are forwarded to a relation extractor.

- **Recall@50% (Digital archive scenario).** In this scenario the relation classifier forwards 50% or more of the pages to the relation extractor with the goal of achieving a high recall. For instance, this scenario may be implemented for a Web search engine or a digital archive scenario, such as [37] [10] [9], where the extraction load can be distributed to many machines. This setting will also be implemented for the Web Analysis Engine [38].

| | Index | | BiGrams | | TriGrams | |
|---|---|---|---|---|---|---|
| Relationship Type | Recall@5 | Recall@50 | Recall@5 | Recall@50 | Recall@5 | Recall@50 |
| Acquisition | 20.8% | 78.3% | 71.5% | **100.%** | 64.9% | **97.5%** |
| CompanyAffiliates | 12.6% | 78.6% | 58.7% | **98.6%** | 70.6% | **96.3%** |
| CompanyEmployeesNumber | 16.7% | 81.9% | 59.2% | **99.5%** | 72.1% | **98.7%** |
| CompanyExpansion | 25.4% | 74.5% | 78.1% | **100.%** | 69.0% | **98.1%** |
| CompanyFounded | 26.9% | 80.3% | 68.0% | **100.%** | 69.3% | **97.5%** |
| CompanyLocation | 12.2% | 70.2% | 59.1% | **100.%** | 61.3% | **97.5%** |
| CompanyProduct | 11.3% | 61.4% | 53.4% | **99.7%** | 62.6% | **97.0%** |
| CompanyReorganization | 15.4% | 88.8% | 82.2% | **98.9%** | 79.9% | **98.9%** |
| CompanyTechnology | 15.3% | 75.8% | 46.1% | **100.%** | 59.3% | **95.6%** |
| CompanyTicker | 8.60% | 72.8% | **100.%** | **100.%** | **95.3%** | **100.%** |
| Conviction | 16.6% | 61.9% | 26.1% | 80.9% | 47.6% | **90.4%** |
| EmploymentRelation | 8.20% | 53.7% | 46.2% | **90.2%** | 41.0% | 86.5% |
| FamilyRelation | 12.0% | 64.8% | 28.0% | **90.1%** | 33.5% | **90.6%** |
| PersonAttributes | 17.5% | 75.5% | 42.7% | 87.7% | 41.9% | 86.2% |
| PersonCareer | 8.99% | 60.0% | 28.9% | **94.0%** | 40.2% | 89.9% |
| PersonCommunication | 13.4% | 67.8% | 29.1% | 86.5% | 54.3% | 88.2% |
| PersonEducation | 18.1% | 84.4% | 33.7% | 83.1% | 33.7% | 85.7% |
| PersonTravel | 12.2% | 82.2% | 34.4% | **96.6%** | 37.7% | 80.0% |

Table 5.3: For each of the 18 different relation extraction pipelines Recall@Costs is presented for Recall levels of 5% and 50% of forwarded pages. The results are shown for the index ordering according to the Search Engine ranks and the bi- and trigram classifiers.

Table 5.3 lists the recall values for these two scenarios: 5% and 50% of processed pages respectively. It compares the results of the two SVM-Models based on bigrams and trigrams to the results obtained when the documents are processed in the order of their Search Engine rank (index).

- **Recall@5% (Ad-hoc query scenario).** The classifiers drastically increase the recall by about factor four for almost all examined relationship types. It is interesting to note that the TriGram Model outperforms the BiGram Model for a couple of relationship types. However the BigramModel already achieves perfect recall of 100% after processing only 5% of the pages in the corpus for the relationship type CompanyTicker.

- **Recall@50% (Digital archive scenario).** The classifier achieve a recall of 90% or higher for 14 out of 18 relationship-types with the BigramModel. For The relationship types Acquisition, CompanyExpansion, CompanyFounded, CompanyLocation, CompanyTechnology and CompanyTicker perfect recall is achieved. All recall levels lie at least above 80%.

## 5.5 Discussion

The evaluation results clearly show the usefulness of the classifiers as relation predictors. They robustly predict relevant pages for 18 different relation extractors.

### Robust prediction for various attribute and relationship types

With our simple and small feature set the predictors recognize the textual representation of values for very different attribute types, such as date, company, person, location, product, product type, technology, conviction charge, ticker symbol, job position, or degree. Moreover, this feature set enables the relation predictors to recognize not just binary relations, but also more complex relations with five or more attribute values in Web text.

Robust prediction for rare and frequent relationship types. The classifiers robustly identify relevant pages for both, frequent and infrequent, relationship types. As a result, they are particularly helpful for 'rare' relationship types. For instance, relations of the type companytechnology, conviction, companyexpansion, or personeduction appear in less than 2% of the pages. The classifiers effectively filter out irrelevant pages for these relationship types.

#### Slightly higher recall for domain 'company'

Even though the classifiers show excellent performance across all relationship types of the domain Company, recall results are slightly but consistently lower for the relationship types of the domain Person. For instance: the classifier achieved a recall of 95% or more for all ten relationship types from the domain company in the Recall@50% scenario. Contrary, measurements for relationship types that contain attribute values of the type person, such as PersonCommunication or Conviction are below 90% in the Recall@50% scenario.

The most likely explanation for this is probably missing anaphora and co-reference resolution. Take the following excerpt as an example:

***President Barack Obama*** *received the Serve America Act after Congress' vote.* ***He*** *signed the bill last Thursday.* ***The president*** *said it would greatly increase service opportunities for the American People.*

In this text fragment there are three distinct mentions of the entity *Barack Obama.* If one is trying to extract the relation *signed(Executive,Legislation)* form the sentence *'He signed the bill last Thursday.'*, the attempt will only be successful if the extractor is capable of co-reference resolution. This means that the extractor has to be able to infer, that *'He'* actually refers to *'Barack Obama'*. Classifiers based on our approach of generalizing sentences are not capable of making those inferences. Furthermore the quality of Entities chosen to generate the corpus turned out to be quite poor.

After examining misclassified pages that contain relations for these types and attribute values, 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 turned out to be another problem.

### 5.5.1 Relevance for Application Scenarios

#### Ad-Hoc Queries over Web Text

Across all relationship types, the recall increases most rapidly for the first couple of processed pages (<5%). A classifier as an relation predictor can thus be of greatest use for the *Ad-hoc query scenario*. An example application for this scenario would be a system that enables SQL-like queries on Web text data.

Consider the following informal query:

*Extract and join relations between products, acquisitions and employee sizes of companies into result tuples using the web as information source.*

This query might be issued by an analyst who is observing the market for mergers and acquisitions. The query intention is retrieving at tuples about any company, its products and its employee information from Web pages.

Discovering such result tuples with structured queries on Web pages is a grand challenge. For instance, current Web search engines are not capable of extracting relations from search results. Due to the nature of the business models of commercial Web search engines, it is unlikely that full access to internal databases will ever be granted to individual users. Typically only the top 1000 results for a given search query can be retrieved. Therefore a likely approach taken by a Web user who is typing to obtain results for the informal query stated above would be the following:

1. type in some more or less arbitrarily chosen keywords related to the relations into a Web search engine
2. read some of the top ranked result pages
3. copy text phrases that potentially represent relevant tuples into a spreadsheet

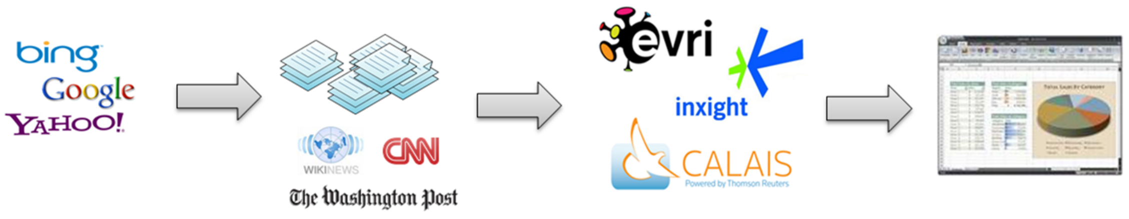


Figure 5.12: An illustration of the Ad-hoc Query Scenario

This problem could also be solved as an ad-hoc retrieval and extraction task. Loeser et al. [39] discuss the problem of supporting analytical business intelligence queries over web-based textual content. Commercial Web search engines are leveraged as an index to retrieve relevant documents from news Web pages. Theses documents can then be forwarded to an information extraction service which returns structured tuples that have been extracted from each page. Finally this structured data can be aggregated in the form of business intelligence reports. Figure 5.12 illustrates this approach.

But even when one applies automatic keyword generation strategies such as the one proposed in [3], many of the retrieved web pages do not contain a relation. To produce results within an acceptable time frame for ad-hoc use, a relation predictor to filter out irrelevant pages is thus a crucial component of any ad-hoc analytics application.

### Analytic Search Engine GoOlap

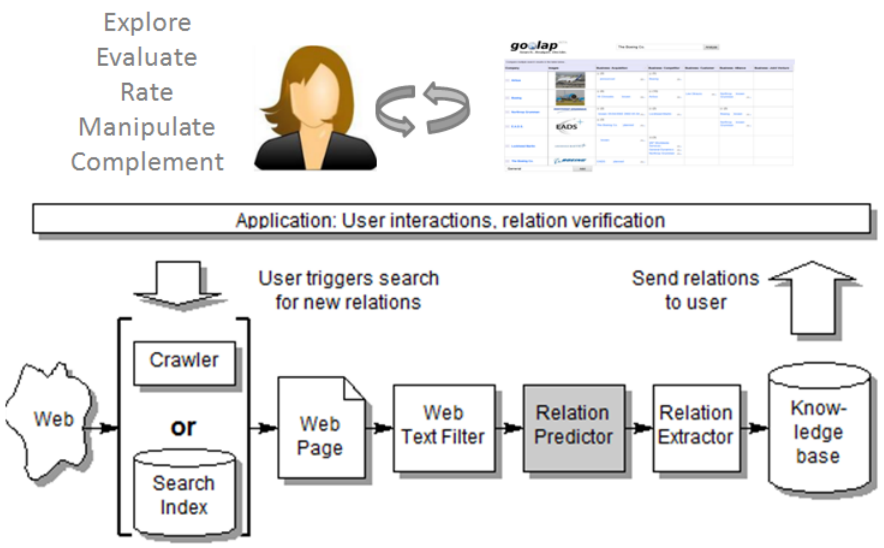


Figure 5.13: A user may discover that some relations are missing in an analytical search engine such as goolap.info. In this case the users request may trigger the entire process of knowledge discovery from the Web.

Another application scenario is an analytic web search engine such as `goolap.info` [38]. GoOlap enables a user to search for entities. It presents a structured and aggregated overview of facts about that entity provided by a knowledge base.

Missing facts or entities in this knowledge base can be retrieved leveraging the infrastructure just described for the ad-hoc query application. One could imagine a scenario where missing facts in the knowledgebase are retrieved online, triggered by the users actions for example. Figure 5.13 illustrates this process.

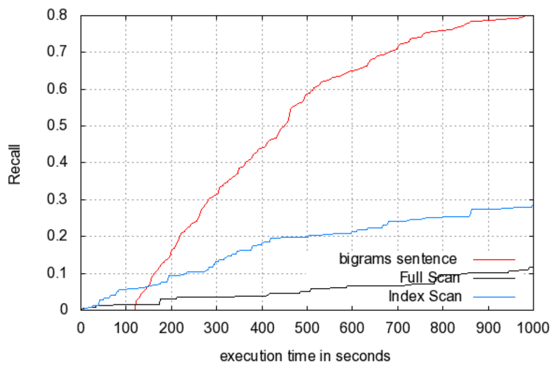
The relation predictor could also play a critical role in this process of knowledge discovery. This application setting actually bears resemblance to the digital archive scenario discussed earlier, since the system actually aims for nearly perfect recall. The relation predictor could speed up the system and conserve processing resources by effectively filtering out irrelevant pages during the knowledge discovery process.

### 5.5.2 Classification Function for Ranking Pages

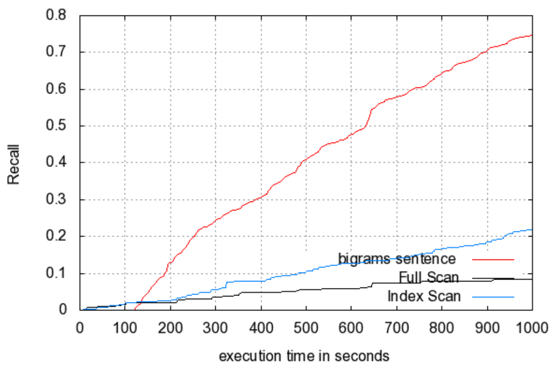
If the relation predictor where to be used for an Ad-hoc Extraction application, the threshold would have to be set a priori. This might turn out to be a difficult parameter to set because it is not intuitive for a user. It would be much more desirable to control the execution time and simpley cut-off the retrieval and extraction process after a given time limit.

To enable this functionality the documents have to be ordered according to some ranking function. This is a similar approach as the Extraction Prioritization proposed by the authors of [24]. The classification function already assigns a function value $y$ to every document. The greater it is, the more confident the classification algorithm is, that the page in question contains an extractable fact. It thus seems reasonable to uses the classification function to rank the documents. For that to happen however, all the documents have to be evaluated by the classification function first.

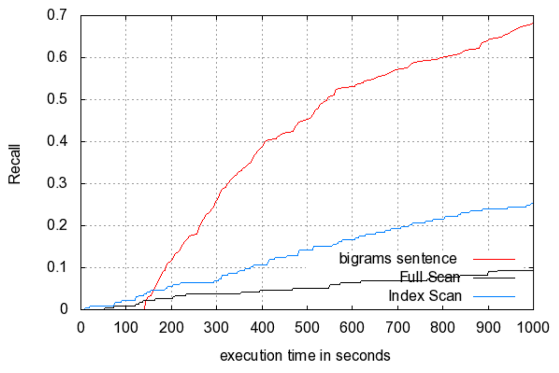
To illustrate the performance of such an approach, the resulting recall curves have been generated for a max runtime of 1000 seconds. The results are displayed in Figures 5.14 through 5.14.



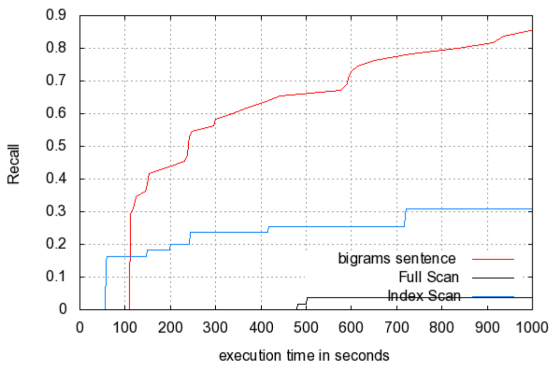
(a) Recall Curves for acquisition



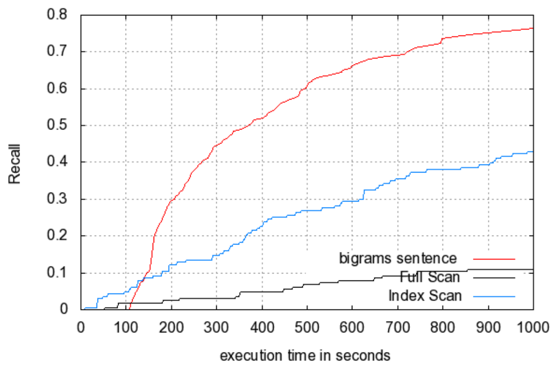
(b) Recall Curves for companyaffiliates



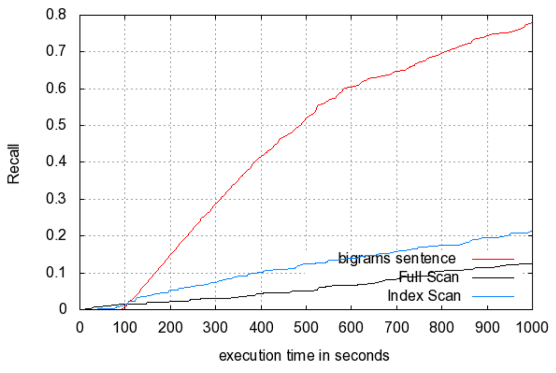
(c) Recall Curves for companyemployeesnumber



(d) Recall Curves for companyexpansion

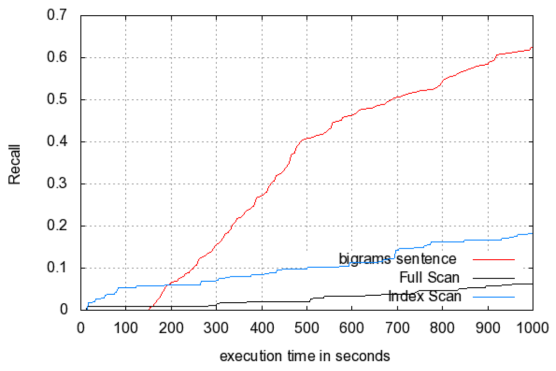


(e) Recall Curves for companyfounded

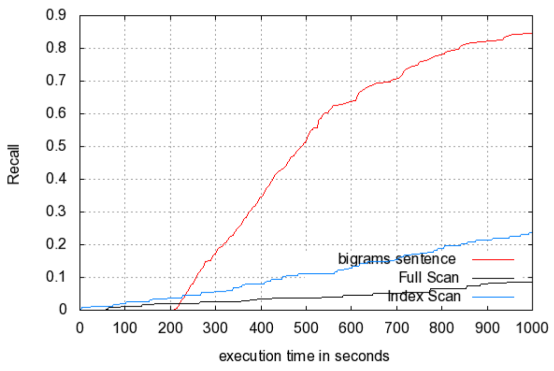


(f) Recall Curves for companylocation

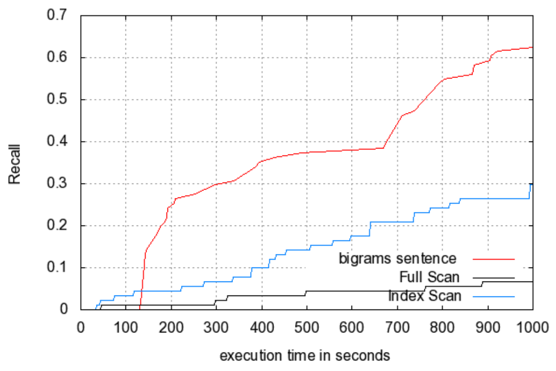
Figure 5.14: Recall vs. Execution time for the first 1000 seconds Part I



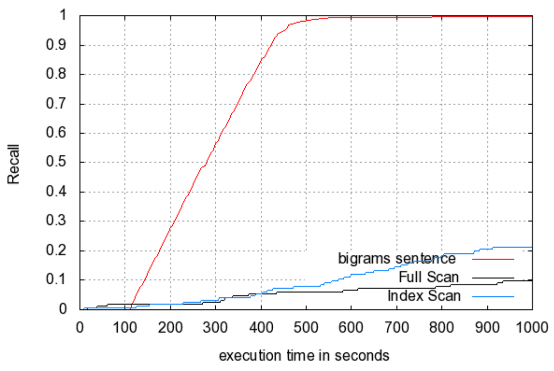
(a) Recall Curves for companyproduct



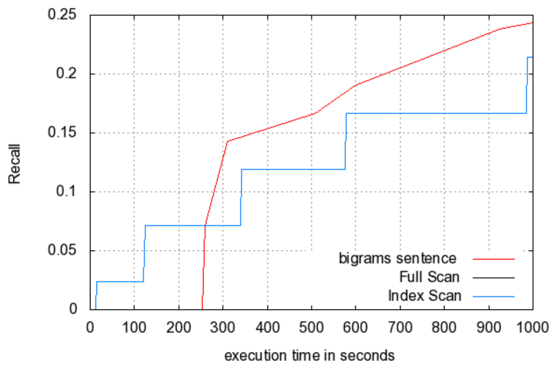
(b) Recall Curves for companyreorganization



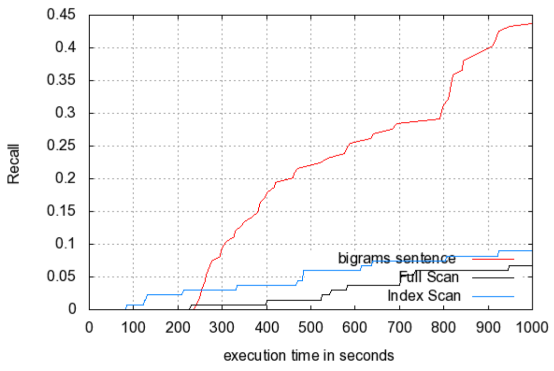
(c) Recall Curves for companytechnology



(d) Recall Curves for companyticker

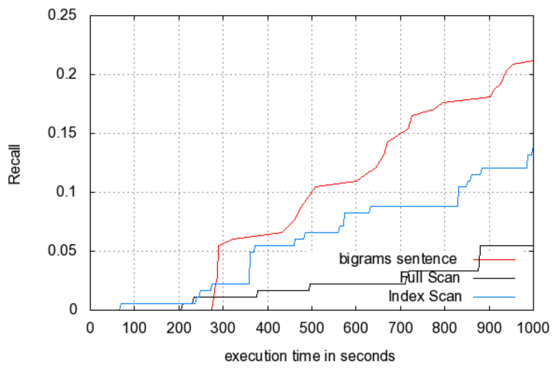


(e) Recall Curves for conviction

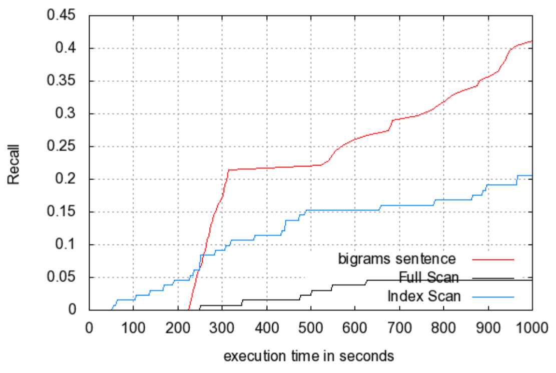


(f) Recall Curves for employmentrelation

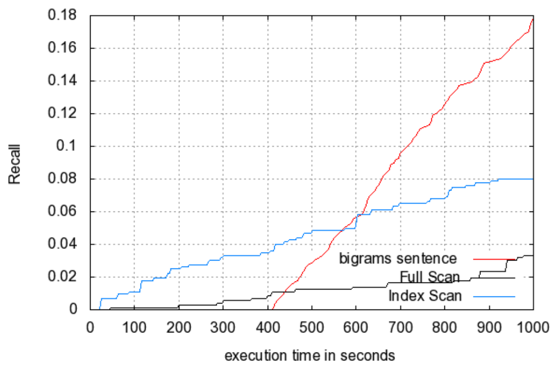
Figure 5.15: Recall vs. Execution time for the first 1000 seconds Part II



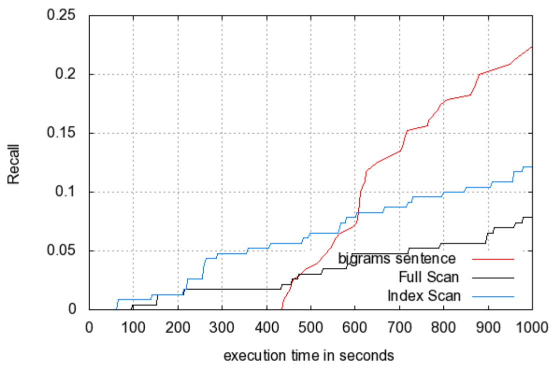
(a) Recall Curves for familyrelation



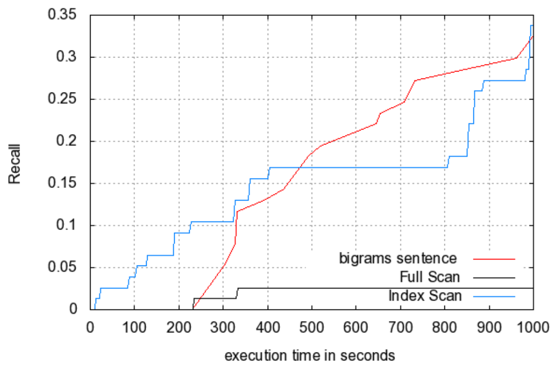
(b) Recall Curves for personattributes



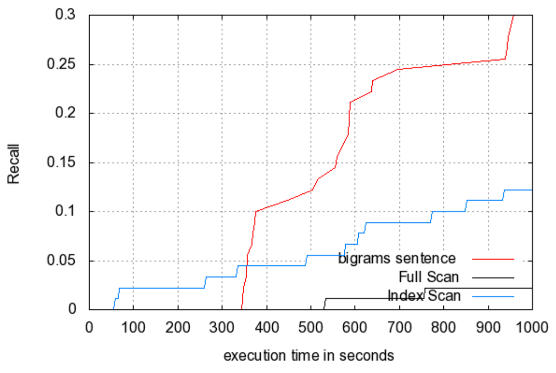
(c) Recall Curves for personcareer



(d) Recall Curves for personcommunication



(e) Recall Curves for personeducation



(f) Recall Curves for persontravel

Figure 5.16: Recall vs. Execution time for the first 1000 seconds Part III

Since the classifier first has to evaluate all documents to compute the classification function value, no facts are extracted in the first couple of seconds. Note however that this evaluation was performed over the entire corpus, so the system had to evaluate between 500 and 7500 documents before extraction could begin. In a more refined scenario one could limit the number of pages as well.

Even though it is evident, that using the relation predictor for ranking the documents according to their classification function value significantly outperforms the approach of processing the documents in the order of their search engine rank. (Recall that the search engine has already been queried with relationship-type specific, discriminative keywords.) If this classifier ranking where to be used in the 'Ad-Hoc Queries over Web Text' application scenario, the user could simply set a time-out or stop the extraction process at any time and view the results that have been retrieved so far.

## 5.6 Summary

To be able to consistently train and evaluate classification algorithms, we generated a fixed corpus of annotated web Pages for 18 different relationship-types from the domains Company and Person.

We retrieved the pages by querying a commercial Web search engine with relation specific keyword queries generated with the strategy described in [3]. The Web pages were sent to an extraction service to generate 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.

In a broad experimental evaluation on more than hundred thousand pages I evaluated both bigram and trigram based support vector machines for all of the 18 different relation extractors. The prediction performance has been measured by generation ROC Curves and AUC estimates by classifying unseen test data through 10 fold cross-validation.

The evaluation results clearly show the usefulness of the svm classifiers as relation predictors. They robustly predict relevant pages for 18 different relation extractors. The predictors are fast and maintain their performance across the examined domains. On average they are about is about two orders of magnitude faster than the actual relation extraction service itself. With our simple and small feature set the predictor recognizes the textual representation of values for very different attribute types.

The classifiers would be particularly useful in an *Ad-hoc Query Scenario* where only few pages can be processed due to tight time constrains. The recall curves show that they classifiers cause a substantial increase in recall for this scenario. However the algorithm could also be useful for the knowledge discovery process of analytical web search engines such as `goolap.info`.

# 6 Conclusion

## 6.1 Summary

Scaling information extraction to large document collections, such as the Web, is a challenging problem. Current relation extraction systems are computationally expensive, e.g., they might require several seconds to process a single page. Therefore it is infeasible to sequentially scan and forward all Web pages to the extractor. Often such an exhaustive inspection of all pages might not be necessary, since only a few relevant pages contain a textual representation of a relation.

In this thesis, we presented a *relation predictor*, which classifies Web pages as either relevant or irrelevant for a relation extraction task. Our approach is inspired by the observation that most binary relationships in the English language are expressed using only a rather small set of patterns. We developed a method to generalize individual words to a set of shallow text, syntactic and semantic tags where each sentence of a document is transformed into a token representation of linguistic features.

As a classification algorithm, we used linear Support Vector Machines (SVMs) based on bi- and trigrams.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. Our relation predictor helps to deploy existing relation extraction systems at a large scale and for a wider range of applications than previously possible.

The evaluation results clearly show the usefulness of the svm classifiers as relation predictors. They robustly predict relevant pages for 18 different relation extractors. The predictors are fast and maintain their performance across the examined domains. On average they are about is about two orders of magnitude faster than the actual relation extraction service itself. With our simple and small feature set the predictor recognizes the textual representation of values for very different attribute types.

The classifiers would be particularly useful in an *Ad-hoc Query Scenario* where only few pages can be processed due to tight time constrains. The recall curves show that they classifiers cause a substantial increase in recall for this scenario. However the algorithm could also be useful for the knowledge discovery process of analytical web search engines such as `goolap.info`.

## 6.2 Future Work

As future work we plan to deploy the relation predictor for the analytical Web search engine goolap.info and in prototypical applications for processing Ad-Hoc Querys over Web text. But besides deployment of the developped component, there are also two interesting research challenges worth exploring:

**Ranking documents with Classification functions** While the idea to use the classification function such as an svm to learn a ranking function has already been proposed and examined e.g., by Joachims [40], it would be interesting explore the idea further in the context of ranking pages for information extraction.

In chapter 5 we already surveyed how the classification function itself could be used to rank the documents for information extraction. It would be an interesting to explore this idea further and to try to find an optimal ranking function for the information extraction task.

**Language Models and a Naive Bayes Classifier** As mentioned in Chapter 3 there are multiple ways to extract and encode *n-grams* from text or our token representation. One of these possibilities is to apply *Language Models.*

These models estimate the probability of words given the previous $n - 1$ tokens:

$$P(w_n | w_1, \ldots, w_{n-1})$$

The authors of [41] propose to use language models to augment a Naive Bayes Classifier to a so called 'Chain Augmented Naive Bayes'. They used this classifier to perform text classification and report promising results.

It would be interesting to explore how a generative model such as the Chain Augmented Naive Bayes Classifier would perform when applied to the task of relation prediction. The probabilistic judgements $P(DocumentContainsRelation|Features)$ could also be used to rank the documents as has been shown for the SVM classifier in Chapter 5.

# Bibliography

[1] Haefele T. Loeser A. Boden, C., “Classification algorithms for relation prediction”, in *DaLi Workshop at ICDE 2011 (forthcomming)*, 2011.

[2] Eugene Agichtein and Luis Gravano, “Querying text databases for efficient information extraction”, in *In Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE*, 2003, pp. 113–124.

[3] Nagel C. Pieper S. Loeser, A., “Augmenting tables by self-supervised web search”, in *BIRTE Workshop at VLDB 2010*, 2010.

[4] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl, “Boilerplate detection using shallow text features”, in *Proceedings of the third ACM international conference on Web search and data mining*, New York, NY, USA, 2010, WSDM ’10, pp. 441–450, ACM.

[5] Erik F. Tjong Kim Sang and Fien De Meulder, “Introduction to the conll-2003 shared task: language-independent named entity recognition”, in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, Morristown, NJ, USA, 2003, pp. 142–147, Association for Computational Linguistics.

[6] Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom, “Coreference resolution with reconcile”, in *Proceedings of the ACL 2010 Conference Short Papers*, Morristown, NJ, USA, 2010, ACLShort ’10, pp. 156–161, Association for Computational Linguistics.

[7] Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning, “A multi-pass sieve for coreference resolution”, in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Morristown, NJ, USA, 2010, EMNLP ’10, pp. 492–501, Association for Computational Linguistics.

[8] Einat Amitay, Nadav Har'El, Ron Sivan, and Aya Soffer, "Web-a-where: geotagging web content", in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2004, SIGIR '04, pp. 273–280, ACM.

[9] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld, "Open information extraction from the web", *Commun. ACM*, vol. 51, pp. 68–74, December 2008.

[10] Fei Wu and Daniel S. Weld, "Open information extraction using wikipedia", in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Morristown, NJ, USA, 2010, ACL '10, pp. 118–127, Association for Computational Linguistics.

[11] Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni, "Semantic role labeling for open information extraction", in *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, Morristown, NJ, USA, 2010, FAM-LbR '10, pp. 52–60, Association for Computational Linguistics.

[12] Ronen Feldman, Yizhar Regev, and Maya Gorodetsky, "A modular information extraction system", *Intell. Data Anal.*, vol. 12, pp. 51–71, January 2008.

[13] Vasin Punyakanok, Dan Roth, and Wen-tau Yih, "The importance of syntactic parsing and inference in semantic role labeling", *Comput. Linguist.*, vol. 34, pp. 257–287, June 2008.

[14] Nancy Chinchor, *Proceedings of the Seventh Message Understanding Conference*, Science Applications International Corporation (SAIC), San Francisco, CA, 1998.

[15] Nancy Chinchor, "Muc-4 evaluation metrics", in *Proceedings of the 4th conference on Message understanding*, Morristown, NJ, USA, 1992, MUC4 '92, pp. 22–29, Association for Computational Linguistics.

[16] Ellen Riloff, "Automatically generating extraction patterns from untagged text", in *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996, pp. 1044–1049.

[17] Soumen Chakrabarti, Sunita Sarawagi, and S. Sudarshan, "Enhancing search with structure", *IEEE Data Eng. Bull.*, vol. 33, no. 1, pp. 3–24, 2010.

[18] Ralph Grishman, Silja Huttunen, and Roman Yangarber, "Information extraction for enhanced access to disease outbreak reports", *J. of Biomedical Informatics*, vol. 35, pp. 236–246, August 2002.

[19] Raphael Hoffmann, Congle Zhang, and Daniel S. Weld, "Learning 5000 relational extractors", in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Morristown, NJ, USA, 2010, ACL '10, pp. 286–295, Association for Computational Linguistics.

[20] Panagiotis G. Ipeirotis, Eugene Agichtein, Pranay Jain, and Luis Gravano, "To search or to crawl?: towards a query optimizer for text-centric tasks", in *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, New York, NY, USA, 2006, SIGMOD '06, pp. 265–276, ACM.

[21] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, and Hongjun Lu, "Discriminative category matching: Efficient text classification for huge document collections", in *ICDM*. 2002, pp. 187–194, IEEE Computer Society.

[22] "Opencalais", `http://www.opencalais.com/documentation/calais-web-service-api/api-metadata/entity-index-and-definitions`, (Last visited 01/10/10).

[23] Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick R. Reiss, and Shivakumar Vaithyanathan, "Systemt: an algebraic approach to declarative information extraction", in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Morristown, NJ, USA, 2010, ACL '10, pp. 128–137, Association for Computational Linguistics.

[24] Jian Huang and Cong Yu, "Prioritization of domain-specific web information extraction", in *AAAI*, Maria Fox and David Poole, Eds. 2010, AAAI Press.

[25] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld, "Open information extraction from the web", *Commun. ACM*, vol. 51, pp. 68–74, December 2008.

[26] Razvan C. Bunescu, "Learning to extract relations from the web using minimal supervision", in *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL '07*, 2007.

[27] "Yahoo boss", `http://developer.yahoo.com/search/boss/`, (Last visited 01/10/10).

[28] Alias-i., "Lingpipe 4.0.1.", http://alias-i.com/lingpipe, (Last visited 01/10/10).

[29] Thorsten Joachims, "Text categorization with support vector machines: learning with many relevant features", in *Proceedings of ECML-98, 10th European Conference on Machine Learning*, Claire Nédellec and Céline Rouveirol, Eds., Heidelberg et al., 1998, pp. 137–142, Springer.

[30] Vladimir N. Vapnik, *The nature of statistical learning theory*, Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[31] Bernhard Schölkopf and Alexander J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*, The MIT Press, 1st edition, December 2001.

[32] Michael Jordan, "Figure taken from: Classification lecture", `http://www.cs.berkeley.edu/~jordan/courses/294-fall09/lectures/classification/`, (Last visited 01/10/10).

[33] "Figure: Receiver operating characteristic", `http://en.wikipedia.org/wiki/File:ROC_space-2.png`, (Last visited 01/10/10).

[34] Andrew P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms", *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, July 1997.

[35] Jin Huang and Charles X. Ling, "Using auc and accuracy in evaluating learning algorithms", *IEEE Trans. on Knowl. and Data Eng.*, vol. 17, pp. 299–310, March 2005.

[36] Alpa Jain and Divesh Srivastava, "Exploring a few good tuples from text databases", in *Proceedings of the 2009 IEEE International Conference on Data Engineering*, Washington, DC, USA, 2009, pp. 616–627, IEEE Computer Society.

[37] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor, “Freebase: a collaboratively created graph database for structuring human knowledge”, in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, New York, NY, USA, 2008, SIGMOD ’08, pp. 1247–1250, ACM.

[38] “Goolap”, `http://www.goolap.info/`, (Last visited 01/10/10).

[39] Alexander Lser, Steffen Lutter, Patrick Dssel, and Volker Markl, “Ad-hoc queries over document collections  a case study”, in *Enabling Real-Time Business Intelligence*, vol. 41 of *Lecture Notes in Business Information Processing*, pp. 50–65. Springer Berlin Heidelberg, 2010.

[40] T. Joachims, “Evaluating Retrieval Performance Using Clickthrough Data”, 2002.

[41] Fuchun Peng and Dale Schuurmans, “Combining Naive Bayes and n-Gram Language Models for Text Classification”, *Lecture Notes in Computer Science*, vol. 2633, pp. 335–350, January 2003.