Report on current state-of-the-art methods for relationship mining
The INDECT Consortium

AGH – University of Science and Technology, AGH, Poland
Gdansk University of Technology, GUT, Poland
InnoTec DATA GmbH & Co. KG, INNOTEC, Germany
IP Grenoble (Ensimag), INP, France
MSWiA¹ - General Headquarters of Police (Polish Police), GHP, Poland
Moviquity, MOVIQUITY, Spain
Products and Systems of Information Technology, PSI, Germany
Police Service of Northern Ireland, PSNI, United Kingdom
Poznan University of Technology, PUT, Poland
Universidad Carlos III de Madrid, UC3M, Spain
Technical University of Sofia, TU-SOFIA, Bulgaria
University of Wuppertal, BUW, Germany
University of York, UoY, Great Britain
Technical University of Ostrava, VSB, Czech Republic
Technical University of Kosice, TUKE, Slovakia
X-Art Pro Division G.m.b.H., X-art, Austria
Fachhochschule Technikum Wien, FHTW, Austria

© Copyright 2009, the Members of the INDECT Consortium

¹ MSWiA (Ministerstwo Spraw Wewnętrznych i Administracji) – Ministry of Interior Affairs and Administration. Polish Police is dependent on the Ministry
## Document Information

<table>
<thead>
<tr>
<th>Contract Number</th>
<th>218086</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable name</td>
<td>Report on current state-of-the-art methods for relationship mining</td>
</tr>
<tr>
<td>Deliverable number</td>
<td>4.2</td>
</tr>
<tr>
<td>Editor(s)</td>
<td>Suresh Manandhar, University of York, <a href="mailto:suresh@cs.york.ac.uk">suresh@cs.york.ac.uk</a></td>
</tr>
<tr>
<td>Author(s)</td>
<td>Ioannis P. Klapaftis, University of York, <a href="mailto:giannis@cs.york.ac.uk">giannis@cs.york.ac.uk</a></td>
</tr>
<tr>
<td>Reviewer(s)</td>
<td>Nick Pears, University of York, <a href="mailto:nep@cs.york.ac.uk">nep@cs.york.ac.uk</a></td>
</tr>
<tr>
<td>Dissemination level</td>
<td>Public</td>
</tr>
<tr>
<td>Contractual date of delivery</td>
<td>30/10/2009</td>
</tr>
<tr>
<td>Delivery date</td>
<td>30/10/2009</td>
</tr>
<tr>
<td>Status</td>
<td>Final version</td>
</tr>
<tr>
<td>Keywords</td>
<td>Relation extraction, Information extraction</td>
</tr>
</tbody>
</table>

This project is funded under 7th Framework Program
Report on current state-of-the-art methods for relationship mining
Table of Contents

Executive Summary .................................................................................................................. 8

Introduction ............................................................................................................................. 9

1.1 Objectives ....................................................................................................................... 9

1.2 List of participants & roles .............................................................................................. 9

1.3 The task of Relation Extraction ....................................................................................... 10

1.4 RE Evaluation setting ..................................................................................................... 11

1.5 RE applications & relation to INDECT .......................................................................... 12

1.6 Overview & structure of the report .................................................................................. 13

Supervised Methods for Relation Extraction ......................................................................... 14

2.1 Generative Models .......................................................................................................... 14

2.2 Discriminative Models .................................................................................................... 15

2.3 Conditional Random Fields ............................................................................................ 16

2.3.1 CRFs for Relation Extraction ..................................................................................... 18

2.4 Support Vector Machines .............................................................................................. 20

2.4.1 SVMs for Relation Extraction .................................................................................... 23

2.5 Summary ........................................................................................................................ 28

Weakly-supervised approaches .............................................................................................. 30

3.1 Pattern & syntax-based approaches .................................................................................. 30

3.2 Summary ........................................................................................................................ 35

Unsupervised approaches ...................................................................................................... 37

4.1 Distributional clustering approaches ............................................................................... 37

© INDECT Consortium – www.indect-project.eu
4.2 Methods based on hypothesis testing .................................................................42

4.3 Summary..............................................................................................................45

Conclusions..............................................................................................................46

References..............................................................................................................48

Document Updates..................................................................................................52
(This page is left blank intentionally)
Executive Summary

Security is becoming a weak point of energy and communications infrastructures, commercial stores, conference centers, airports and sites with high person traffic in general. Practically any crowded place is vulnerable, and the risks should be controlled and minimized as much as possible. Access control and rapid response to potential dangers are properties that every security system for such environments should have. The INDECT project is aiming to develop new tools and techniques that will help the potential end users in improving their methods for crime detection and prevention thereby offering more security to the citizens of the European Union.

In the context of the INDECT project, Work Package 4 (WP4) is responsible for the Extraction of Information for Crime Prevention by Combining Web Derived Knowledge and Unstructured Data. This document is the second deliverable of WP4 and provides a detailed review of the current-state-of-the-art in the field of relationship mining. In this survey, current work is organised into three categories. The limitations of each category are identified and discussed, in effect providing the basis for overcoming those in the next stages of the project.
Introduction

The general aim of WP4 is the development of key technologies that facilitate the building of an intelligence gathering system by combining and extending the current-state-of-the-art methods in Natural Language Processing (NLP). One of the specific goals of WP4 is to propose NLP and machine learning methods that learn relationships between people and organizations through websites and social networks. Key requirement for the development of such methods is the analysis of the current-state-of-the-art methods for Relation Extraction (RE).

1.1 Objectives

In this report, we provide a critical survey of the field of relation extraction. Based on our review, relation extraction methods are divided into three categories, i.e. supervised, weakly-supervised and unsupervised approaches.

This categorization is based on the ability of approaches to overcome the knowledge acquisition bottleneck. This common problem appears in most areas of NLP and it is caused by the lack of adequate training data for building classifiers using learning methods. These classifiers are then applied to unseen data in order to perform their task e.g. relation extraction, word sense disambiguation, question answering and others.

The objective of this deliverable is to identify the advantages of current supervised, weakly supervised and unsupervised approaches, as well as their limitations, which we aim to address in the next stages of project by developing novel relation extraction methods.

1.2 List of participants & roles

This report has been produced by the University of York (UOY), and has been utilized by INNOTEC for the purpose of dissemination (D9.5)
1.3 The task of Relation Extraction

Relationship mining or relation extraction is defined as the task of identifying a relation of interest that exists between two or more entities. For example, consider the following fragments of text.

- **George Brown** is working for **InSoft LTD**.
- **George Brown** is married to **Lilian Brown**.

The target of a RE system is to identify that there is an employment relation between the named entity **George Brown** and the organization **InSoft LTD**. Equivalently a RE system should also identify that there is a family relationship between the entities **George Brown** and **Lilian Brown**.

The above example has considered two different types of binary relation. Most relation extraction systems focus on the specific problem of extracting binary relations and little work has been done in recognising and extracting more complex relations [33]. Complex relations are relations involving more than two entity arguments, i.e. n-ary relations. These relations are more informative than binary ones, as the following example shows.

- **George Brown** is the CEO of **InSoft LTD**.

We observe that the employment relation can now be defined as a ternary relation in which the first argument might be the employee, the second its job title and the third its employer.

The above examples illustrate that in order to perform relation extraction, it is necessary to devise a method that identifies the entities in a given text, as well as their type. For example, a RE system should identify that **George Brown** is a person and that **InSoft LTD** is an organization in order to learn or predict that the former is an employee and the latter is an employer.

Most research on information extraction is based on the discovery of named entities [33]. Early named-entity taggers were based on finite-state generative models [6]. Later on, discriminatively-trained models were applied [28, 32, 48] and shown to be more accurate than generative models. These models have also been applied to relation extraction, as we will see in the next sections of the report.
1.4 RE Evaluation setting

The most common framework for the evaluation of RE systems is based on an information retrieval evaluation setting, i.e. by means of recall, precision and harmonic mean of these [39].

The majority of methods described in the following sections follow this evaluation setting, although depending on the domain and the type of applications some approaches have considered a different evaluation as presented in the next sections.

Let us assume that a given corpus describes $N$ relations between entities. This set of relations is our Gold Standard (GS). Let us also suppose that a system has extracted a total of $M = m_{\text{correct}} + m_{\text{incorrect}}$ relations by applying its methodology to the corpus, where $m_{\text{correct}}$ is the number of correct extracted relations according to the GS, and $m_{\text{incorrect}}$ is the number of incorrect extracted relations.

\[
P = \frac{m_{\text{correct}}}{M} \]

*Equation 1: Precision of a RE system*

Equation 1 defines the precision of the system and Equation 2 defines its recall. Finally, recall and precision are combined to produce F-Score (harmonic mean) in Equation 3.

\[
R = \frac{m_{\text{correct}}}{N} \]

*Equation 2: Recall of a RE system*

Intuitively, precision captures the accuracy of a system by measuring the percentage of the extracted relations that are actually correct. In contrast, recall assesses the coverage of system by measuring the percentage of the extracted relations in the GS that have been captured by the system.

\[
F\text{-Score} = \frac{2 \times P \times R}{P + R} \]

*Equation 3: Harmonic mean of the recall & precision of a RE system*
1.5 RE applications & relation to INDECT

The importance of relation extraction results from its potential applications. In particular, RE can be used to improve the accuracy of search engines by allowing them to handle complex queries, i.e. queries whose correct answers depend on their semantic interpretation.

For example, current search engines are unable to answer accurately queries (Figure 1), such as the following:

- What is the relationship between Barack Obama and Natasha Obama?

A search engine that is able to exploit relations between entities can semantically interpret the above query and return results that mention Barack Obama as the father of Natasha Obama.

In the same vein, RE can also be used to enhance the performance of Question/Answering (QA) systems. For example, by treating the date as a type of entity and introducing a birth date relation, a QA system exploiting these relations can answer questions such as “When was Y born?”.

Initial applications of this approach can be found in [36], which developed a method for downloading from the web textual patterns, e.g. patterns which included a named
entity and a date, \( X \) was born in \( Y \), where \( X \) is an entity and \( Y \) is a date. These patterns can be interpreted as simple relations between entities and dates.

Bioinformatics is another area where RE has many potential uses. For example, within the context of the advanced NLP and Text Mining project [1], RE has been used to extract relations between genes and different forms of cancer. In this application, both genes and diseases need to be treated as entities. The massive amount of medical literature that has been developed throughout the years can then be processed and exploited by a RE system that will identify these entities and their relations.

Within the context of INDECT, relation extraction is a key factor, because it allows the detection of unknown relationships between people (e.g. known criminals) or between people and organizations from unrestricted corpora such as blogs, forums and other internet resources. These relationships can further be exploited by the police partners or by an automated system that aims at the automatic detection of threats and recognition of abnormal behaviour or violence.

Additionally, the relationships extracted by a RE system are important features that can be exploited by methods aiming at the behavioural profiling of known criminals. Given that the relation types in which an entity participates are strongly reflected on his/her behavioural patterns, a behavioural profiling method should also consider this type of information in order to build behavioural patterns.

1.6 Overview & structure of the report

The field of relation extraction can be broadly divided into three categories. The first one includes supervised systems, i.e. systems which exploit annotated data to train a classifier that is then applied to unseen corpora to predict relations between entities. The second category includes systems that attempt to overcome the main limitation of supervised systems, i.e. their portability to domains where annotated data do not exist. Weakly-supervised systems typically exploit a few hand-crafted patterns and then apply a bootstrapping process in order to learn new patterns predicting new relations between entities. Finally, the last category of approaches includes unsupervised systems. Typically, these systems cluster pairs of entities that share the same relation without making use of any hand-tagged data.

We begin our review by first presenting supervised methods and then we describe weakly-supervised and unsupervised approaches. For each of these categories we describe in detail a number of methods and discuss their strengths and weaknesses in the summary of each section.
Supervised Methods for Relation Extraction

A large part of work in relation extraction has focused on the development of supervised approaches. Typically supervised learning uses training data (annotated data), i.e. pairs of input vectors and desired outputs to learn a function, which is then applied to unseen data in order to predict an output.

In the following sections, we first provide a brief overview on two types of supervised learning, and then we focus on specific supervised approaches.

2.1 Generative Models

For the purpose of this section, we assume that we have to deal with a classification task, i.e. to assign a given input to one or more predefined categories based on a set of features that characterise this input. For example, in document classification, we aim to assign a given document to one or more categories based on its contents, so that each category consists of a set of documents with similar subjects. Similarly, in RE, we would like to assign a candidate relation between two or more entities to one relation type or none given the features that characterise this candidate relation.

Formally, let \( Y \) be a random variable for class labels (categories) that takes values \( \{ Y_1, ..., Y_m \} \), and \( X \) another random variable describing an input instance, which is represented as a real-valued vector of \( n \) features \( \{ X_1, ..., X_n \} \). Given an input and its corresponding feature vector, the task of classification consists of computing \( P(Y_i | X_1, ..., X_n) \) for \( i = 1...m \) and then choosing the \( Y_i \) with the highest conditional probability as the label.

These conditional probabilities can be easily calculated using the joint distribution of class labels and features, i.e. \( P(Y_i | X_1, ..., X_n) = \frac{P(Y_i, X_1, ..., X_n)}{P(X_1, ..., X_n)} = \frac{P(Y_i)P(X_1, ..., X_n | Y_i)}{P(X_1, ..., X_n)} \).

However this process requires the computation of every combination of values of the \( n \) features, hence it is impossible to make learning tractable in problems with hundreds of features.

It is straightforward from the previous discussion that we need to make some independence assumptions about the features in order to make learning tractable. The naïve Bayesian categorization assumes that the features of an instance are independent of each other given the class label. Thus, \( P(Y_i, X_1, ..., X_n) \) can be estimated using Equation 4.
Equation 4: Naïve Bayes categorization

Thus, we only need to calculate \( P(X_j \mid Y_i) \) for each pair of feature-value and class label. In the case, where features and labels are binary, we only need to specify \( 2^n \) parameters, as opposed to \( 2^n \) when we do not make independence assumptions. Naïve Bayes belongs to generative models, i.e., models that specify a joint distribution between observations and labels (categories) [47].

2.2 Discriminative Models

Discriminative models [47] are specifically designed to directly model the conditional distribution \( P(Y \mid X) \) hence they are specifically suited for classification tasks. A simple discriminative model is logistic regression used to estimate the probability of an event by fitting the data to a logistic curve [47]. Equation 5 defines the probability of class label 1 given a vector of features in binary logistic regression. Naturally, the probability of class label 0 given the vector of features is \( 1 - P(Y = 1 \mid X) \).

Equation 5: Classification for logistic regression

Logistic regression is basically a linear model. In particular, we can use Equation 6 to support the argument that we assign label 0, if \( \frac{P(Y = 0 \mid X)}{P(Y = 1 \mid X)} < 1 \). Taking the logs of the inequality provides the following:

\[
\log(1) > \log(\exp(w_0 + \sum_{i=1}^{n} w_i X_i)) \iff 0 < w_0 + \sum_{i=1}^{n} w_i X_i
\]

Binary logistic regression associates with each feature \( X_i \) a weight \( w_i \) that denotes the importance of the feature in the classification. Note that \( w_0 \) is a bias weight for this class label. These weights are computed during the training phase. Specifically, given a set \( T \) of training examples (pairs of input vectors and labels), the weights are estimated during training by maximizing the conditional likelihood of the data (Equation 6), where \( Y' \) denotes the label for training example \( t \), \( X' \) denotes its feature vector and \( W \) is the vector of weights. Several training methods can be used to find the parameters including, gradient descent, conjugate gradient and others.

Equation 6: Binary Logistic Regression conditional data likelihood maximization
The binary case of logistic regression can be easily extended to multi-class problems. This is accomplished by defining feature functions for each combination of class and feature, as well as for the bias weight of each class:

- \( f_{y^j} = 1, \text{ if } Y = y', X = j \text{ and } 0 \text{ otherwise.} \)
- \( f_y = 1, \text{ if } Y = y' \text{ and } 0 \text{ otherwise.} \)

The final conditional distribution is defined in Equation 7, where \( \alpha_k \) are weights (estimated during training), \( f_k \) indexes each feature function and \( Z(X) \) is a normalizing constant.

\[
P(Y \mid X) = \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} \alpha_k f_k(Y, X)\right)
\]

*Equation 7: Multinomial Logistic Regression*

Logistic Regression can be visually illustrated as an undirected graphical model over a set of random variables where an edge represents a dependency. This model is also known as a *Markov Net*.

![Figure 2: Logistic Regression as a simple Markov Net](image)

**2.3 Conditional Random Fields**

In the previous section, we have seen that if no assumption is made, then an exponential number of parameters have to be estimated for probabilistic inference. An assumption of conditional independence allows efficient training and makes inference possible. However it can also lead to low performance.

Graphical models use directed or undirected graphs over a set of random variables and allow for less strict independence assumptions while also limiting the number of parameters that have to be estimated. Figure 2 shows an example of an undirected
Conditional Random Fields (CRFs) [47] are discriminative models that have mostly been applied to sequence labelling problems. Sequence labelling is a task that arises in NLP problems such as part-of-speech (POS) tagging, semantic role labelling and others. It has been recently introduced in relation extraction. Figure 3 shows the graphical representation of a CRF, where x’s are features and y’s states of the sequence.

![Graphical representation of a CRF](image)

**Figure 3: Linear-chain CRF**

Modelling the conditional distribution in CRFs is similar to the one introduced above for logistic regression. Specifically, we create feature functions $f_k(Y_t, Y_{t-1}, X_t)$ for each state transition and state observation pair:

- $f_y(Y_t, Y_{t-1}, X_t) = 1$ if $Y_t = i$ and $Y_{t-1} = j$ and 0 otherwise
- $f_{io}(Y_t, Y_{t-1}, X_t) = 1$ if $Y_t = i$ and $X_t = o$ and 0 otherwise

Having defined the feature functions, a simple linear-chain CRF is defined in Equation 8, where $\alpha_k$ are weights and $Z(X)$ is a normalising factor. Labelling an unseen sequence can be done by computing the most likely labelling ($y^* = \arg\max_y p(y \mid x)$) using a variant of the well-known Viterbi algorithm [50] used in Hidden Markov Models (HMMs).

$$P(Y \mid X) = \frac{1}{Z(X)} \exp\left(\sum_{i=1}^{T} \sum_{t=1}^{K} \alpha_k f_k(Y_t, Y_{t-1}, X_t)\right)$$

**Equation 8: Form of the conditional distribution in CRFs**

Sutton and McCallum [47] note that training of CRFs can be achieved in a similar manner to logistic regression, i.e. by maximizing the conditional log-likelihood ($\sum_{i=1}^{N} \log(p(y^{(i)} \mid x^{(i)}))$, where each $x^{(i)}$ is a sequence of observations, each $y^{(i)}$ is a sequence of desired predictions, and $N = \{x^{(i)}, y^{(i)}\}$ are the training data. The conditional log-likelihood function is concave [47], which means that every local
optimum is also a global optimum.

The simplest approach to maximizing the log-likelihood function is to use the method of *steepest ascent*. In this method we start with an arbitrary point  and move up to the maximum of our function . To accomplish this, we take a sequence of steps until we are are satisfied that we are close to the maximum of the function. In each step, we choose the direction in which our function increases most quickly. This is the direction defined by the gradient of our function at a given point.

The method of steepest ascent requires too many iterations to be practical [47]. In particular, steepest ascent often takes steps that are in the same direction as earlier steps. Hence, its convergence is very slow. *Conjugate gradient methods* [52] improve upon steepest ascent by using *conjugated directions*, i.e. directions whose maximization along a line does not spoil the maximization of previous directions. Conjugate gradients allow reaching the desired point in a significantly smaller number of iterations than steepest ascent.

An application of CRFs to shallow parsing in [41] shows the superior performance of *conjugate gradient methods* over steepest ascent on very large problems with around 3.8 million features. A detailed description of the training methods for CRFs is out of the scope of this report. Readers can consult [52] for more information.

### 2.3.1 CRFs for Relation Extraction

In [14] the problem of relation extraction is cast into a sequence labelling problem and linear-chain CRFs are applied to label a sequence of entities and extract relational patterns that improve extraction performance.

Culotta et al. [14] focus on biographical texts that mostly discuss one entity called the *principal entity*. The rest of the entities mentioned in a biographical text are referred to as *secondary entities*. For each secondary entity, the goal is to predict the relation to the principal entity. This problem formulation allows treating relation extraction as a sequence labelling task. In the following labelled example, we observe that the *principal entity* (Barack Obama) is connected to the secondary entities (Michelle Obama, Malia Ann and Natasha Obama). The labels above each secondary entity denote its relation to the principal one.

- **Barack Obama**
  - Spouse
  - Daughter

- **Michelle Obama**
  - Spouse

- **Malia Ann**
  - Daughter

- **Natasha Obama**
  - Daughter

By applying a sequence labelling task, the entity recognition step is folded into the relation extraction task [14]. As a result, there is no need to label each entity as a person or company using an NER tagger. Instead, an entity’s label is its relation to the principal entity. However, is has to be noted that anchoring one argument of relations to be the principal entity adds an extra supervision factor.
The modelling flexibility of CRFs allows the inclusion of many different and possibly overlapping features, which are realised as feature functions as we have seen in the previous sections. The list of local features in [14] includes context words (within a 6 word window around the target secondary entity), lexicon features (whether a token appears in a list of cities, people, etc.), regular expressions (whether a token is capitalised, contains digits, etc), and part-of-speech of surrounding words.

In addition to the local features, they have also included relational pattern features to improve extraction accuracy. The relational pattern features are patterns that connect entities and are automatically learned from a database. An example of pattern is shown in Figure 4.

![Figure 4: Pattern template for siblings relation](image)

The process of relation extraction consists of two stages. In the first stage, a CRF that does not use relational patterns is applied in order to learn a set of relations between entities and build a database. Relational patterns can now be identified from the database. Each extracted relation is assigned a confidence value based on the CRF probabilistic extraction model. In the second stage, further extraction is achieved by a CRF exploiting the already identified relational patterns, i.e. treating these as features.

The method was evaluated on 271 articles from Wikipedia containing a total of 4701 relations that were labelled manually. The data were split into training and test sets (70%-30% split), and the extraction results were measured in terms of recall, precision and harmonic mean.

<table>
<thead>
<tr>
<th></th>
<th>LR</th>
<th>CRF – NRP</th>
<th>CRF – RP1</th>
<th>CRF – RP2</th>
<th>CRF – RP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-Score</td>
<td>54.89</td>
<td>59.95</td>
<td>61.00</td>
<td>60.08</td>
<td>61.36</td>
</tr>
<tr>
<td>Precision</td>
<td>64.75</td>
<td>70.19</td>
<td>67.99</td>
<td><strong>71.77</strong></td>
<td>70.95</td>
</tr>
<tr>
<td>Recall</td>
<td>47.63</td>
<td>52.32</td>
<td><strong>55.31</strong></td>
<td>51.66</td>
<td>54.06</td>
</tr>
</tbody>
</table>

*Table 1: Relation Extraction Evaluation in [14]*

Table 1 shows the results of their evaluation under different settings. LR and CRF-NRP refer to a logistic regression and a CRF classifier respectively. Both of them do not use relational patterns as features. CRF-RP1 is a CRF classifier that incorporates relational patterns as features, while CRF-RP2 and CRF-RP3 filter out the relational pattern features according to their confidence weight as will be explained below.

We observe that CRF-NRP outperforms the logistic regression classifier and demonstrates that modelling the dependence between entities in text is a significant source of information that should be taken into account. Table 1 also shows that CRF-RP1 performs similarly to CRF-NRP. However, it is also clear that the inclusion of patterns provides a significant increase in recall at the cost of a significant decrease in precision.

To deal with this problem, CRF-RP2 and CRF-RP3 filter out the relations extracted in the first stage if their confidence weight is below 0.9 and 0.5 respectively. This strategy would allow for a more accurate extraction of relation patterns from the automatically generated database. CRF-RP3 seems to identify the best compromise between recall and precision, hence providing the best F-Score.

Given that previous results do not clearly show whether the inclusion of relation patterns can offer a significant advantage in performance, they run two more experiments in which the relational pattern features are fixed to be correct (they are taken from the test set). In the first one, all of the correct relational patterns were used, while in the second only half of them. Their results show a significant increase in precision and recall in both experiments confirming the initial intuition that relational features can help extraction. However, the difficulty in capturing correct patterns suggests that a semi-automatic database construction might be more appropriate, which increases the degree of supervision.

### 2.4 Support Vector Machines

Support Vector Machines (SVMs) are supervised learning methods that have been used in different NLP tasks including text classification [22], question classification [55], word sense disambiguation [23, 29], and more recently to relation extraction [54].

SVMs provide a principled approach to classification and provide good generalization [22]. Additionally, SVMs have an explicit dependence on the data (via the support vectors as we will see below); hence they can interpret the generated model. More importantly, SVM learning involves optimization of a concave function, i.e. there are no local minima.

For binary classification, let us assume that we have a set of input vectors \( x_j \). Our target is to predict their labels \( y_j = \pm 1 \). The motivation for using SVMs comes from
theoretical bounds on the generalization error. These bounds have two important implications: (1) the upper bound on the generalization error does not depend on the dimensionality of the space and (2) the bound is minimized by maximizing a quantity called the margin, i.e. the minimal distance between the hyperplane separating the two classes (1, -1) and the closest data points (support vectors) of each class. This is illustrated in Figure 5, where our classifier is a line in two dimensions.

Figure 5: Hyperplane separating 2 classes

In an n-dimensional space a hyperplane has the form \( w \cdot x + b = 0 \), where \( w \) is the vector of weights and \( b \) is the bias factor. Thus, the decision function for binary classification is \( D(x) = \text{sign}(w \cdot x + b) \). Note that the outcome of this function is scale invariant for any positive number \( l \).

This means that we can implicitly fix a scale so that the distance between a support vector and the hyperplane is equal to 1, i.e. \( w \cdot x_i + b = 1 \) for a support vector in the plus side of Figure 5 and \( w \cdot x_j + b = -1 \) for a support vector in the minus side of Figure 5. Thus, \( w \cdot (x_i - x_j) = 2 \) for two support vectors in each side of the classifying hyperplane. The margin can be easily computed by the projection of the vector \((x_i - x_j)\) onto the vector normal to the hyperplane \( \frac{w}{|w|} \), i.e. \( \frac{w}{|w|} (x_i - x_j) \). Since we already know that \( w \cdot (x_i - x_j) = 2 \), we can infer that the margin is equal to \( \frac{2}{|w|} \). Thus maximizing the margin is equivalent to minimizing \( \frac{|w|}{2} \), subject to the constraint \( y_i (w x + b) \geq 1 \), which results from the previous implicit fixing of the scale.

This constraint maximization problem can formulated as finding the optimum of the
Following primal objective function: \( L(w,b,\alpha) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{m} \alpha_i [y_i ((w \cdot x_i) + b) - 1] \), where \( \alpha_i \) are Lagrange multipliers. Typically, it is more convenient working with the dual form of \( L(w,b,\alpha) \), which can obtained by solving the saddle point equations \( \frac{\partial L}{\partial b} = 0 \) and \( \frac{\partial L}{\partial w} = 0 \). The dual form of \( L(w,b,\alpha) \) is shown in Equation 9. This dual form is the function that we wish to maximize. Equation 9 defines a quadratic function, and hence SVMs can be trained using conjugated gradient methods as CRFs.

\[
W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)
\]

*Equation 9: Dual function of \( L(w,b,\alpha) \)*

Finally for binary classification the decision function is shown in Equation 10, where \( q \) is new test instance.

\[
D(q) = \text{sign} \left[ \sum_{i=1}^{m} \alpha_i y_j (x_i \cdot q) + b \right]
\]

*Equation 10: SVM Decision function for binary classification*

All of the above discussion has assumed that our data points can be separated. However, in many cases this might not be true or we might not even know whether the data points are separable. SVMs exploit the second implication of the generalization error, i.e. the theoretical bounds do not rely on dimensionality of the space. This in effect means that we can map our data points to a higher dimensionality space where more of the data points are separable.

By observing the dual objective function, we can see that our datapoints are in the form of an inner product; therefore to get a better representation of the data we can replace our inner product with another defined in another higher dimensional space \( x_i \cdot x_j \rightarrow T(x_i) \cdot T(x_j) \), where \( T(x_i) \) is the mapping function. The function \( K(x_i, x_j) = T(x_i) \cdot T(x_j) \) is called a kernel and defines the inner product between mapped pairs in the high dimensional space. In fact, one does not need a priori knowledge of the functional form of the mapping function as this will implicitly be defined by the functional form of the mapped inner product.

Different kernels can be used depending on the application of interest, e.g. Gaussian kernel [46], string kernels [30], graph kernels [27] as well as dependency tree kernels [53, 54]. Naturally, there are restrictions on the use of kernels since not all functions define an inner product in a high dimensional space. A simple criterion is to check whether the kernel function is positive semi-definite.
2.4.1 SVMs for Relation Extraction

Zelenko et al. [53, 54] present an application of kernel methods for extracting relations from text. The novelty in their supervised approach lies in the introduction of a new kernel function for computing the similarity between shallow parse trees. In contrast to full parsing, shallow parsing [4] does not provide the full interpretation of a sentence, although it extracts its key elements. Thus, shallow parsing is robust and able to identify structured representations even for sentences that are not grammatically correct.

Given the sentence “George Brown is a well-known scientist at York University”, the shallow parser produces the tree shown in Figure 6, where we observe that there are three nodes referring to the person George Brown, one with type Person and two with type PNP. Also there is an Organization entity, i.e. York University.

Given the shallow parse tree of Figure 6, one can extract person-organization relations by taking a person or PNP node, and one organization node from the tree and then assigning the Role attributes to each one of them. An example is shown in Figure 7. From the tree of Figure 6 we can also extract two more positive relation examples by instantiating each time one of the PNP nodes and keeping the Organization fixed.

Formally, a node in the shallow parse tree is a set of attributes \( \{a_1, a_2, ..., a_n\} \). The attributes are named, and each node has the Role and Type attributes defined. Given the definition of a node, a relation example can be defined inductively [54] as

\[ Type = Sentence \]

\[ Type = Person \]
Text = George Smith

\[ Type = Verb \]
Head = be

\[ Type = PNP \]
Head = scientist

\[ Type = PNP \]
Head = scientist

\[ Type = Prep \]
Text = at

\[ Type = Entity \]
Text = York University

\[ Type = DET \]
Text = a

\[ Type = ADJ \]
Text = well-known

\[ Type = Noun \]
Head = scientist

*Figure 6: Shallow parse tree example. (The type PNP refers to Personal Noun Phrase)*
follows:

- Given a node \( p \), the pair \( P=(p,[]) \) is a relation example, where \([\] \) is an empty sequence.
- Given a node \( p \) and a sequence of pairs \( [P_1, P_2, \ldots, P_n] \), then the pair \( P=(p,[P_1, P_2, \ldots, P_n]) \) is a relation example.

\[
\text{Type} = \text{Sentence} \\
\text{Role} = \text{none}
\]

\[
\begin{align*}
\text{Type} &= \text{Person} \\
\text{Text} &= \text{George Smith} \\
\text{Role} &= \text{member}
\end{align*}
\]

\[
\begin{align*}
\text{Type} &= \text{Verb} \\
\text{Head} &= \text{be} \\
\text{Role} &= \text{none}
\end{align*}
\]

\[
\begin{align*}
\text{Type} &= \text{PNP} \\
\text{Head} &= \text{scientist} \\
\text{Role} &= \text{none}
\end{align*}
\]

\[
\begin{align*}
\text{Type} &= \text{Prep} \\
\text{Text} &= \text{at} \\
\text{Role} &= \text{none}
\end{align*}
\]

\[
\begin{align*}
\text{Type} &= \text{Entity} \\
\text{Text} &= \text{York University} \\
\text{Role} &= \text{Organization}
\end{align*}
\]

**Figure 7: Person-Organization relation example**

A relation example can be labelled as positive (+1) if it depicts the target relation between the instantiated target entities. Otherwise, it is labelled as negative (-1). Also \( P . p \) denotes the first element (parent) of the relation example, and \( P . c \) denotes the second element (children).

Having these definitions, the first step in applying SVMs for relation extraction is to define a kernel function, i.e. function that represents the similarity of two shallow subtrees. Zelenko et al. first define a kernel function on the nodes in terms of their attributes and then they extend it on relation examples. Specifically, given two relation examples \( P_1 \) and \( P_2 \), the function \( K(P_1, P_2) \) is a kernel function which is equal to 0, if \( P_1.p \) and \( P_2.p \) are incompatible, i.e. they have different \( \text{Type} \) and \( \text{Role} \). If they are compatible, then \( K(P_1, P_2) = k(P_1.p, P_2.p) + K_c(P_1.c, P_2.c) \), where \( k(P_1.p, P_2.p) \) is 1 if the text attributes of \( P_1.p \) and \( P_2.p \) are identical and 0 if not. \( K_c(P_1.c, P_2.c) \) is the similarity function of the children of \( P_1.p \) and \( P_2.p \).

Zelenko et al. [54] present two versions of \( K_c(P_1.c, P_2.c) \), one which operates on contiguous subtrees and a second operating on sparse subtrees. The kernel function for contiguous subtree kernels is shown in Equation 11, where \( i, j \) are children sequences of \( P_1 \) and \( P_2 \), \( l(i) \) is the length of sequence \( i \), and \( t(P_1[i], p, P_2[j], p) \) is a function comparing the nodes of two sub-sequences in terms of their compatibility.
(type and role attributes) text of their parents. When the text of two parents are compatible \( t(P_1[i], p, P_2[j], p) \) returns 1, otherwise it returns 0.

\[
K_c(P_1, c, P_2, c) = \sum_{i, j \in c(i)} K(P_1[i], P_2[j]) \prod_{x = I \cup J} t(P_1[i], p, P_2[j], p)
\]

**Equation 11 : Similarity function for contiguous tree kernels**

Intuitively, the previous equation enumerates all sub-sequences of relation example children with matching parents, adds the corresponding child examples similarities, while decreasing this similarity by a factor reflecting how spread the sub-sequences within children sequences. For sparse subtrees, the computation is similar with the difference that the sequences do not have to be contiguous.

The method was evaluated on two types of relations, i.e. person-organization and organization-location. The corpus consisted of 200 news articles. These were shallow parsed, and then relations were extracted and manually labelled as positive or negative. 60% of the corpus was used as training set and the rest as testing. SVMs with the kernel functions along with the voted perceptron method [19] and naïve Bayes were used for training and testing. Performance was measured in terms of recall and precision over 10 random train/text splits.

<table>
<thead>
<tr>
<th></th>
<th>Recall</th>
<th>Precision</th>
<th>F-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Bayes</td>
<td>75.59</td>
<td>91.88</td>
<td>82.93</td>
</tr>
<tr>
<td>Voted Perceptron (contiguous)</td>
<td>79.58</td>
<td>89.74</td>
<td>84.34</td>
</tr>
<tr>
<td>Voted Perceptron (sparse)</td>
<td>81.62</td>
<td>90.05</td>
<td>85.61</td>
</tr>
<tr>
<td>SVM (feature-based)</td>
<td>76.21</td>
<td>91.67</td>
<td>83.22</td>
</tr>
<tr>
<td>SVM (contiguous)</td>
<td>79.78</td>
<td>89.9</td>
<td>84.52</td>
</tr>
<tr>
<td>SVM (sparse)</td>
<td>82.73</td>
<td>91.32</td>
<td>86.8</td>
</tr>
</tbody>
</table>

**Table 2 : Person-affiliation performance in [53,54]**

Tables 2 and 3 show the results of their evaluation in the two relation types. In Tables 2, 3 SVM (contiguous) refers to an SVM with contiguous subtree kernels and SVM (sparse) refer to an SVM with sparse subtree kernels. The same applies for the voted perceptron. SVM (feature-based) refers to an SVM, whose features correspond to small subtrees of the shallow parse representations of relation examples [54]. The exact definition of these features is given in the appendix of Zelenko et al. [54].

Overall, we first observe that SVMs perform better than naïve Bayes and their equivalent voted perceptron. Secondly, the feature-based SVM, which does not include dependency tree kernels, performs worse than the SVM with sparse kernels and the SVM with contiguous kernels. This result demonstrates that syntactic dependencies provide an important source of structural information that cannot be
captured by bag-of-words and other type of features.

\begin{center}
\begin{tabular}{|l|c|c|c|}
\hline
 & Recall & Precision & F-Score \\
\hline
Naïve Bayes & 71.94 & 80.40 & 80.04 \\
Voted Perceptron (contiguous) & 64.43 & 92.85 & 76.02 \\
Voted Perceptron (sparse) & 71 & 91.9 & 80.05 \\
SVM (feature-based) & 70.32 & 88.18 & 78.17 \\
SVM (contiguous) & 71.43 & 92.03 & 80.39 \\
SVM (sparse) & 76.33 & 91.78 & 83.3 \\
\hline
\end{tabular}
\end{center}

Table 3: Organization-Location performance in [53,54]

Additionally, their results show that SVMs with sparse tree kernel outperform SVMs with contiguous tree kernels in both target relations. This in effect indicates that sparse tree kernels are able to deal effectively with data sparsity and improve upon the strict contiguous kernels.

Culotta & Sorensen [13] have developed a similar approach that instead uses augmented dependency trees to define kernel functions. Assuming that named entity tagging has been applied, all pairs of entity pairs within the same sentence are gathered and full parsing is performed in each sentence. A parse tree is then transformed to a dependency tree, which denotes the grammatical relations between words, e.g. subject are dependent on their verbs and adjectives are dependent on the nouns, they modify [13].

Given an entity pair, the lowest common subtree is extracted from the original dependency tree. Culotta & Sorensen [13] note that this extraction is applied in order to reduce noise and give emphasis on local characteristics of relations. Each node of the dependency subtree is then augmented with a set of features including its token, its part-of-speech, its chunk, its WordNet hypernyms and others.

Formally, let $T$ be a tree with nodes $\{t_1, \ldots, t_n\}$, where each node is associated with a vector of features $\Phi(t_i) = \{v_{i_1}, \ldots, v_{i_m}\}$. The kernel function between two trees is defined in Equation 12, where $m(r_1, r_2)$ is a binary matching function that determines whether the roots of the two subtrees are compatible. In their method, the matching function included three features, i.e. the POS tag of the root, its entity type and its relation argument.

$$
K(T_1, T_2) = \begin{cases} 
\{s(r_1, r_2) + K_c(r_1[c] + r_2[c]) \text{ if } m (r_1, r_2) > 0 \} \\
\{0 \text{ if } m (r_1, r_2) \}
\end{cases}
$$

Equation 12: Kernel function in Culotta & Sorensen [13]

The function $s(r_1, r_2)$ is a similarity function between two nodes which is defined in
Equation 13, where $C(v_k, v_q)$ equals 1 if $v_k = v_q$ and 0 otherwise.

$$s(t_i, t_j) = \sum_{v_k \in \Phi(t_i)} \sum_{v_q \in \Phi(t_j)} C(v_k, v_q)$$

*Equation 13: Similarity function between feature vectors in Culotta & Sorensen [13]*

Finally, $K(r[c] + r[c])$ is the kernel function over children defined as in Zelenko et al. [53] (see page 25). Intuitively, whenever a pair of matching nodes is found, all matching subsequences are enumerated. For each matching pair of nodes in a matching subsequence the result of function $s(t_i, t_j)$ is accumulated, and recursion is applied to matching subsequences of their children.

The method was evaluated on the ACE dataset that consists of 800 annotated documents. The target entities were Person, Organization, Geo-political entity, Location, and Facility. There were 24 relation types between these entities.

Additionally, they first used a binary SVM classifier to detect possible relations, and then they use a multiclass SVM to classify each predicted relation. Their results are presented in Table 4, where column D refers to the type of SVM used for detection and column C refers to the type of SVM used for classification.

Culotta & Sorensen [13] have experimented both with contiguous, SVM(contiguous) and sparse tree kernels, SVM(sparse), as well as with a simple bag-of-words kernel, SVM(BoW), in which the features are binary (either a word exists or not). Although SVM training was performed on the 24 relations, their testing was only applied on 5 high-level relation types. Two more kernels were developed, i.e. SVM (BoW + sparse) and SVM(BoW + contiguous).

<table>
<thead>
<tr>
<th>D</th>
<th>C</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (sparse)</td>
<td>SVM (sparse)</td>
<td>66.0</td>
<td>29.0</td>
<td>40.1</td>
</tr>
<tr>
<td>SVM (contiguous)</td>
<td>SVM (contiguous)</td>
<td>66.6</td>
<td>32.4</td>
<td>43.5</td>
</tr>
<tr>
<td>SVM (BoW)</td>
<td>SVM (BoW)</td>
<td>62.5</td>
<td>27.7</td>
<td>38.1</td>
</tr>
<tr>
<td>SVM (BoW+sparse)</td>
<td>SVM (BoW+sparse)</td>
<td>67.5</td>
<td>34.3</td>
<td>45.3</td>
</tr>
<tr>
<td>SVM (BoW + contiguous)</td>
<td>SVM (BoW + contiguous)</td>
<td>67.1</td>
<td>35.0</td>
<td>45.8</td>
</tr>
<tr>
<td>SVM (contiguous)</td>
<td>SVM (BoW + contiguous)</td>
<td>67.4</td>
<td>33.9</td>
<td>45.0</td>
</tr>
<tr>
<td>SVM (BoW + contiguous)</td>
<td>SVM (contiguous)</td>
<td>65.3</td>
<td>32.5</td>
<td>43.3</td>
</tr>
</tbody>
</table>

*Table 4: Evaluation Results in [13]*
Table 4 shows the results of their evaluation. We first observe that the bag-of-words kernel has the worst performance, and that generally dependency tree kernels are superior. This verifies the results of Zelenko et al. [53, 54] presented earlier and confirms that syntactic dependencies are more useful for relation extraction.

Interestingly, we also see that the SVM(BoW + contiguous), i.e. the combination of the bag-of-words kernel with the contiguous dependency tree kernel, has the best performance. This in effect shows that the bag-of-words kernel captures information not identified by dependency tree kernels, i.e. co-occurring information. As a result their combination is able to improve both of them.

Finally, their evaluation shows that contiguous tree kernels perform better than sparse tree kernels. This contradicts with Zelenko et al. [53], hence it is unclear which one is better.

2.5 Summary

This section has provided a detailed review of supervised approaches to the problem of relation extraction. Specifically, we have presented and contrasted two types of learning approaches, i.e. generative and discriminative models. The former model the joint distribution of random variables while the latter model their conditional distribution. Their main difference is that in discriminative models the conditional distribution does not include a model of the marginal p(x) which is not needed for classification anyway. The difficulty in modelling p(x) is that it often contains many highly dependent features that are difficult to model while retaining tractability. On the other hand making independence assumption (as in naïve Bayes) might hurt performance. However, it has to be mentioned that the naïve Bayes performs surprisingly well in document classification, although it performs worse on average across a range of applications than logistic regression [8, 54]. To this end, discriminative models appear to be more robust in terms of performance than generative models, mainly due to the fact that they are remain agnostic of the marginal distribution p(x) focusing only on the conditional.

In relation extraction, two types of discriminative models have been applied - conditional random fields and support vector machines with different kernel functions. Their evaluation shows that they perform better than logistic regression, however no comparisons between SVMs and CRFs were found.

The main strength of CRFs comes from the fact that they can model different and possibly overlapping features (via feature functions), hence relax the hard assumption of naïve Bayes. SVMs on the other hand are able to map the data points to a higher dimension (via kernel functions) while at the same time being resistant to overfitting.

Within the context of INDECT, it is crucial to be able to continuously capture and update relations between entities through web blogs, chats and forums within different domains (hooliganism, terrorism and others). As a result, the adoption of supervised techniques would impose restrictions on the portability of the developed
algorithms as well as on the feasibility of their applications, since there are no hand-tagged data for every domain or type of relation.

Furthermore, all of the supervised approaches assume that a target relation is given, which is not always realistic, since new types of explicit or implicit relations between entities may come up. To deal with these limitations, it is important to investigate weakly-supervised or unsupervised methods. An alternative solution that is gaining interest in other areas of NLP, would be to automatically create (high-quality) annotated data, which can then be used to train classifiers using supervised methods.
Weakly-supervised approaches

Relation extraction has traditionally relied on extensive human involvement of hand-crafted rules and training examples [17]. This process is labour-intensive, costly, error-prone and unable to deal with the vast amount of relations that exist in general-purpose corpora, e.g. on the web.

Weakly supervised methods attempt to overcome these deficiencies by removing the need for hand-tagged corpora. Typically, these approaches require a few initial seeds of relation examples as well as the types of the target relations the initial seeds describe. New instances of target relations are then retrieved by applying bootstrapping techniques.

3.1 Pattern & syntax-based approaches

KnowItAll [17] is a web-based information extraction system that exploits a set of predefined patterns and relation types in order to learn a new set of relations between two or more entities.

Specifically, KnowItAll uses lexico-syntactic patterns [21] that suggest that two entities can be engaged in a relation. An example and an instantiation of a lexico-syntactic pattern is the following:

- Pattern: NP is employed by NP
- Instantiation: George Brown is employed by InqSoft LTD

The above pattern defines an employment relationship with two arguments, a person and an organization. By employing Part-Of-Speech (POS) tagging in a given text, one can identify noun phrases and then apply the pattern, in order to retrieve the portions of text which instantiate that pattern. For the sake of precision, one can also restrict the above pattern by replacing the NP's with proper names.

Given a binary target relation type and a set of matching patterns, KnowItAll creates a query for each pattern and each focus class, where focus class is one of the arguments in the relation. For example, if the target relation is employment with arguments employer, employee and the focus class is InqSoft, then the generated query would be the following:

- “is employed by InqSoft”

The constructed queries are sent to a number of search engines including Google, Altavista and others, and the top n documents are retrieved from each engine and downloaded. In the next step, the documents are POS-tagged and the patterns are applied in order to extract a set of instantiated patterns that denote the target relation.

One of the shortcomings of KnowItAll is the fact that it requires the relation types as well as the patterns for each relation type to be provided as an input. Hence,
practically KnowItAll will not be portable across different domains where new relations have to be defined.

In contrast TextRunner [5] is an information extraction system that overcomes this limitation of KnowItAll. TextRunner consists of three key modules:

- Self-supervised Learner
- Single-Pass Extractor
- Redundancy-Based Assessor

The first module operates in two steps. Initially, it labels its own training data as positive or negative. The training data is used to train a naïve Bayes classifier used by the Extractor module. Extracted relations are of the form of a tuple \( t = (e_i, r_i, e_j) \), where the first and third arguments are entities engaged into the relation denoted by the second argument. The Learner module uses a parser [26] in order to syntactically analyse each sentence in the training corpus, and identify noun phrases. The module then traverses the parse structure to identify a sequence of words connecting these noun phrases (entities). This sequence is used to denote the relation \( r_i \).

For example given the following sentence, noun phrases **George Brown** and **InqSoft LTD** would be identified as the first and third argument of the candidate relation respectively. Using the Link Grammar Parser [43, 44], we can see that the candidate relation can be denoted by the verb phrase \( (VP \text{ is } (VP \text{ employed } (PP \text{ by}))) \).

- **George Brown is employed by InqSoft LTD.**

Given an extracted tuple \( t = (e_i, r_i, e_j) \), the Learner uses a set of heuristics in order to label it as a positive or negative example. The heuristics mentioned in the literature are the following:

- The dependency chain between the two entities is no longer than a certain length.
- The path from one entity to another does not cross a sentence-like boundary.
- Neither \( e_i \) nor \( e_j \) consist solely of a pronoun.

Once the Learner has labelled a tuple as positive or negative, it then transforms it into a feature vector representation. The features used are domain-independent and include the sequence of POS tags, the number of tokens in a relation, the number of stop words in the relation and others. The constructed feature vectors and their labels are used to train a naïve Bayes classifier.

The Single-Pass Extractor module is used to process a new document and produce the relations that exist in that document. To accomplish its task, it first identifies noun phrases using a noun phrase chunker, and then produces a set of possible relations (tuples) connecting the noun phrases in the text. The candidate tuples are presented to the classifier, which decides whether they are trustworthy or not.
The third module, Redundancy-Based Assessor, assigns to each retained tuple a probability that depicts the confidence that the relation between the two entities of the tuple is true. The assignment is based on probabilistic model of text redundancy [13].

Another pattern-based approach is presented in [7], where the method of DIPRE (Dual Iterative Pattern Expansion) is introduced. DIPRE is a bootstrapping process, whose input is a set of relation examples and a pattern, which matches these examples in a given text. An example of a pattern would be X's headquarters are in Y, where X is an organization and Y is a location. Note however that DIPRE does not use an NER tagger to identify entities. Instead, it uses simple POS tagging to identify noun phrases.

DIPRE identifies text surrounding the initial relation examples' entities. The identified portions of text are converted to patterns, which are then applied to identify new tuples. These tuples are then used as new seeds in the extraction method. The whole process operates for a predefined number of iterations.

SnowBall [2] is a similar pattern-based approach that attempts to improve DIPRE in terms of precision. This is accomplished by assigning confidence weights both to patterns that generate the tuples and to the tuples used as new seeds to generate the patterns.

The confidence of a pattern is defined in Equation 14, where \( P(\text{positives}) \) refers to the number of tuples, known from previous iterations of the system that are correct. In contrast, \( P(\text{negatives}) \) is the number of tuples that conflict with a tuple extracted from previous iterations. For example, if a previous iteration extracted the tuple <IngSoft, New York> and the current iteration extracts the tuple that InqSoft has its headquarters in London (<InqSoft, London>), then the extracted tuple is considered as a negative one.

\[
Conf(P) = \frac{P(\text{positives})}{P(\text{positives}) + P(\text{negatives})}
\]

*Equation 14: Confidence of a pattern in SnowBall*

Patterns are generated by clustering the matched tuples using a simple single-linkage algorithm. The similarity between two tuples is defined by a Match function that calculates the inner product between the vectors of the two tuples.

A vector for a tuple is created by considering the frequencies of words that appear \( n \) tokens to the left of the first entity, \( n \) tokens to the right of the second entity, and the tokens in the middle of the two entities. The confidence for a tuple \( T \) is defined as a function of the confidence of the patterns that suggested \( T \) and the degree of match of context \( C_i \), associated with an occurrence of the tuple that matched \( P_i \).
Their evaluation on a large collection of articles from the North American News Text Corpus shows that their precision remains stable and their recall increases as the number of iterations increases. In contrast DIPRE’s precision decreases as a result of the inclusion of noisy patterns and tuples.

A similar approach that focuses on learning patterns indicating relations was introduced in [36]. As in the previous approaches, in [36] they initially use a set of predefined relation examples, whose arguments (entities) are sent to a search engine (Google).

Snippets are downloaded and sentences that contain both arguments of the initial binary relations are extracted. Each sentence passes through a suffix tree constructor which finds all substrings of all lengths along with their frequency. For example consider the following sentences:

- Albert Einstein (1879-1955) was a great scientist.
- Einstein (1879-1955) was a genius.
- Einstein (1879-1955) had a rich scientific career.

The longest matching substring is Einstein (1879-1955) and has a count equal to 3. Finally, each extracted substring is only retained, if all the arguments of the binary relation exist.

A thorough evaluation of the above method was reported in [35], where they used the TREC-9 and CHEM datasets. The former consists of a sample of newswire articles, which contain 5951432 words. The second is a small dataset of 313590 words, and is related to chemistry.

Their evaluation shows that the pattern-based learning approach is able to achieve high coverage (recall), but low precision. This means that the generated patterns are generic, hence they can recognise a large number of relation instances. However, these patterns also generate relation instances which are false positives.

To deal with that deficiency Pantel & Pennachiotti [35] presented Espresso, a bootstrapping method that assigns reliability weights both to patterns and extracted instances of relations. The main advantage of their approach as compared to SnowBall [2] is that their method of assigning weights builds upon a principled statistical model, instead of ad-hoc and intuitive statistical measures.

Specifically, Espresso iterates between three phases, i.e. pattern induction, pattern ranking/selection, and instance extraction. During the first phase, the algorithm of [36] (presented above) is applied in order to extract a set of candidate patterns. In the second stage, each pattern is assigned a reliability weight. Only the top \(k\) patterns are retained, so that in each iteration the total number of patterns is equal to the
number of patterns in the previous iteration plus one.

In the third stage, Espresso retrieves from the corpus the set of instances that match any of the accepted patterns. Each instance is also assigned a reliability weight, and then the $m$ highest scoring instances are retained. In their experiments $m$ was set to 200. In the following description we present the statistical model used for assigning weights to patterns and extracted instances.

The idea for assigning reliability weights to patterns comes from the fact that reliable patterns are those that generate highly precise instances. Patterns generating many relation instances are more likely to have a low precision. Hence, a reliable pattern is a pattern having a high association with its matched instances. This association is calculated using the pointwise mutual information [40], a commonly used metric for measuring the strength of association of two events.

\[
  r_p(p) = \frac{\sum_{i \in I} \frac{pmi(i, p)}{\max_{\text{pum}}}}{|I|}
\]

*Equation 16: Reliability of a pattern in Espresso*

Equation 16 defines the reliability of a pattern, where $pmi(i, p)$ defines the similarity between pattern $P$ and instance $i$, $r_i(i)$ is the reliability of instance $i$, $|I|$ is the total number of relation instances, and $\max_{\text{pum}}$ is the maximum PMI between all patterns and instances. Thus, the reliability of a pattern is the average of association across each relation instance, weighted by the reliability of each instance.

Similarly, the reliability of instance $i$ is defined in Equation 17, where $r_p(p)$ is the reliability of pattern $p$, and $|P|$ denotes the total number of patterns at the current iteration. Note that $r_p(p)$ and $r_i(i)$ are recursively defined, where in the first iteration, $r_i(i) = 1$, since the initial relations are provided manually.

\[
  r_i(i) = \frac{\sum_{p \in P} \frac{pmi(i, p)}{\max_{\text{pum}}} \times r_p(p)}{|P|}
\]

*Equation 17: Reliability of a relation instance in Espresso*

Finally, the PMI between an instance $i$ and a pattern $p$ is defined in Equation 18, where the nominator denotes the number of times an instance $i$ with arguments $x, y$ matches the pattern $p$, and the denominator is a product of the total frequency of the pattern (total number of instances it matches) and the frequency of the instance under all patterns.
Equation 18: 
Pointwise mutual information between an instance $i$ and a pattern $p$ in Espresso

$$pmi(i, p) = \frac{|x, p, y|}{|x^*, y^*, p^*|}$$

Additionally in [35], they also attempt to capture generic patterns, and the correct instances they generate. Generic patterns are those, which have a very high recall and a low precision. In their setting, a pattern is classified as being a generic one, if it matches more than 10 times the instances of reliable patterns (patterns accepted at the previous iterations).

An instance of a reliable pattern is assigned a confidence value, which will be higher when many reliable patterns suggest that instance, and at the same time its association with these patterns is high. Equation 19 defines this intuition, where $T$ is the sum of reliability scores of all patterns.

$$S(i) = \sum_{p \in R} S_p(i) \cdot \frac{R_p(i)}{T}$$

Equation 19: Confidence of an instance generated by a generic pattern

Their evaluation on the TREC-9 and CHEM datasets without using generic patterns shows that they are able to achieve a significantly higher precision, than the method of [36]. However, this comes at the cost of having a low recall.

The inclusion of generic patterns increases their recall by two orders of magnitude, while losing on average below 10% precision. This means that the recall climbs at much faster rate than precision decreases [35], and illustrates the fact that the scoring functions effectively separate the correct and incorrect relation instances of generic patterns.

3.2 Summary

Weakly-supervised approaches attempt to overcome the main limitation of supervised method, i.e. the lack of training data for different types of relations and domains. As a consequence, they limit the amount of supervision by requiring two types of inputs: (1) the relations types, and (2) a set of initial seeds describing the relation types.

Earlier weakly-supervised methods focused on applying bootstrapping methods to learn new relation instances and new relation patterns on very large corpora such as the web. This strategy has the advantage of achieving high recall, and the limitation of performing poorly in terms of precision. In other words, the generated patterns are generic enough to capture a large part of relation instances, but also general enough to include false positives.

This limitation of initial weakly-supervised methods was dealt with the introduction of
statistical methods and models that assign weights to extracted relation patterns and relation instances. These weights reflect the reliability or confidence of patterns or relation instances; hence they can be used to filter out noise. The results in [35] show that their weighting mechanism does improve on precision at a faster rate than hurting recall. This in effect indicates that hybrid methods based on both linguistic and statistical knowledge are a viable solution for the purposes of INDECT.

However *data sparsity* is a primary concern for all statistical methods. Data sparsity prevents assigning weights to unseen events or assigning reliable weights to events whose frequency is very low. In general, when dealing with text corpora, data sparsity is always an issue [56].

The methods described so far disregard data sparsity, although this might have a significant effect on performance. This problem is generally dealt with smoothing methods that exploit the distributional similarity of words in text in order to both assign weights to unseen events and improve the unreliability of maximum likelihood estimates for less frequent events.
Unsupervised approaches

Unsupervised approaches aim to overcome the primary limitation of supervised approaches namely the need for richly annotated corpora. Additionally, they do not require initial seeds of relation instances or the types of target relations, in effect improving upon weakly-supervised approaches.

Typically, unsupervised methods of relation extraction apply clustering algorithms in order to group pairs of entities to the same cluster. The label of the cluster defines the target relation, which is shared among all entity pairs of that cluster. As a result, unsupervised approaches have to deal with an even more challenging problem the labelling of the relation.

4.1 Distributional clustering approaches

Hasegawa et al. [20] have proposed an unsupervised method for relation extraction based on context clustering of pairs of entities. Their main assumption is that entity pairs occurring in similar contexts share the same relation hence they can be grouped into the same cluster.

The first stage in their process is the tagging of named entities that exist in a given corpus. This process is accomplished by the use of the OAK NER tagger [40] that labels named entities with their corresponding types, e.g. Person, Organization, Location, and others.

Once the target corpus has been tagged entity pairs are collected. Two entities are considered as a pair if they co-occur in the same sentence and are separated by at most \( n \) intervening words.

Each collected entity pair is associated with a vector whose features are the intervening words between the two entities. These words are stemmed and weighed using TF.IDF (Term Frequency - Inverse Document Frequency). The term frequency for a stemmed word \( w_i \) is the number of co-occurrences in the contexts of the entity pair. IDF is the ratio of the total number of detected entity pairs to the number of entities pairs, in whose contexts \( w_i \) appears.

Finally, the resulting vectors are given as an input to a Hierarchical Clustering Algorithm [24, 45] in which the similarity measure is cosine (Equation 20).

\[
\cosine(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\| \|v_2\|}
\]

Equation 20: Cosine similarity between two vectors
Given a set of singleton clusters $C$ (vectors), HAC produces a binary tree $(L;E)$ with $L$ representing clusters (subsets of $C$) and $E$ representing inclusion relations between clusters of $L$. Initially, each vector belongs to its own cluster and $L$ is set equal to $C$. In subsequent iterations HAC groups the two most similar clusters into a new cluster. The new cluster is added to $L$ and $C$, while the other two are removed from $C$. This process continues until we end up with one cluster in $C$, which is taken to be the root. The following pseudocode describes the process of HAC.

1: Input: A set of vectors, $C = \{c_1, ..., c_m\}$
2: Output: A binary tree with a set of $L$ clusters ($|L| = 2^{|m|} - 1$), and a set of $E$ edges ($|E| = 2^{|m|} - 1$)
3: $L = C$
4: $E = \{\}$
5: $L = m + 1$
6: while $|C| > 1$ do
7:   $(c_a, c_b) = \text{arg max}_{c_i, c_j \in C} \text{sim}(c_i, c_j)$
8:   $C = C \setminus \{c_a\}$
9:   $C = C \setminus \{c_b\}$
10:  $c_n = c_a \cup c_b$
11:  $L = L \cup c_n$
12:  $C = C \cup c_n$
13:  $E = E \cup \{(c_a, c_n)\}$
14:  $E = E \cup \{(c_b, c_n)\}$
15: end while
16: Return $L, E$

For example let us assume that we have the following four sentences.

- George Brown was employed by IngSoft LTD year ago.
- George Brown has been a developer of InqSoft LTD for many years.
- James Smith is employed by Micro.
- James Smith works as a developer for Micro.

Applying NER tagging would provide us with two unique entity pairs, i.e. **George Brown – InqSoft LTD** and **James Smith – Micro**. Each entity pair can now be associated with a vector of features. In our example, the following vectors can be created by only keeping lemmatized nouns and verbs, excluding stop words. Note that each feature is weighed according to its co-occurrence frequency with the entity pair.
We observe that the two vectors are quite similar, which indicates that they could be clustered using HAC. The newly formed cluster would contain both George Brown – InqSoft LTD and James Smith – Micro vectors, and could possibly be merged with other vectors or clusters describing an employment relationship between entity pairs.

An important parameter of agglomerative clustering is the choice of the technique for calculating cluster similarities. Note that as we move towards the higher levels of the taxonomy, clusters contain many feature vectors; hence the choice of the similarity function is crucial as it affects the quality of the clustering solution and the relation extraction performance. Hasegawa et al. [20] have chosen Complete-linkage as their similarity function. Complete-linkage defines as the similarity between two clusters $C_i$ and $C_j$ the minimum similarity among all the pairs of their corresponding vectors, i.e.

$$completeLinkageSimFunction(C_i, C_j) = \min_{v \in C_i, q \in C_j} sim(v, q),$$

where $sim(v, q)$ is the similarity measure between two vectors (cosine in our case). Complete-linkage employs a non-local criterion, since it takes into account the diameter of the candidate merging, i.e. the distance between the two vectors which are furthest from each other. As a result it offers compact clusters. However, it is sensitive to outliers.

For labelling the clusters of HAC, i.e. labelling the relations, they adopted a simple heuristic that labels each cluster with the most frequent features (words) in the entity pair vectors.

Their system was evaluated on The New York Times 1995 corpus, and specifically on two types of relations, PERSON – Geopolitical Entity and Company – Company. They created their gold standard (GS) by manually selecting the entity pairs which participate in each of the given relations.

In the next step, they manually assigned one of the two relation labels in their produced clusters (by observing the pairs), and counted the NE pairs whose relation as defined by the cluster was the same as their relation in the gold standard $N_{correct}$. The pairs on clusters which were not correct were also counted $N_{incorrect}$. Recall was then defined as the ratio of $N_{correct}$ to the total number of relations in their GS, while precision was defined as the ratio of $N_{correct}$ to the total number of relation pairs identified by the system ($N_{correct} + N_{incorrect}$).

They report an 80% F-Score for the PERSON – Geopolitical Entity relation and 75% for the Company – Company. However, they do not provide a comparison against weakly-supervised methods whose problems they aim to overcome.
They also discuss the fact that many of the incorrect groupings of entity pairs was caused by data sparsity. Additionally, they have spotted another problem using distributional similarity approaches, i.e. data sparsity leads to missing interesting yet less frequent pairs of entities. Interestingly, they also mention that their experiments with more common HAC versions (Single-linkage and Average-linkage) yielded worse performance.

A similar approach to the above is described by Shinya & Sekine [42]. The main difference is that clustering does not group entity pairs, but articles based on a set of syntactic features. These syntactic features are connected to entities, hence the final clustering also represents a binary grouping of the entities mentioned in the articles. The obvious disadvantage of such an approach is that relations between entities within a given document are not detected.

Specifically, Shinya & Sekine [42] initially applied a web crawling method which downloaded news web pages for a period of 2 months. They then applied a simple clustering algorithm to find a set of articles that talk about exactly the same news; hence they can form a basic cluster [42]. In their setting, each article was associated with a vector of features, where features were the words of the articles weighed with a variation of TF.IDF. They used cosine similarity between two article vectors, and then they clustered articles from the same day. Their clustering computed the similarity between all pairs of articles and selected the pairs with the highest similarity (threshold 0.65) to form a basic cluster.

The news articles within the basic clusters were syntactically analyzed using a statistical parser [9] and a rule-based tree normaliser. In parallel with the syntactic analysis of each basic cluster, they also used a NE tagger [40] to label named entities with their corresponding types. Next, entities among different articles in a basic cluster were connected to obtain cross-document co-reference entities.

Each unique entity of a basic cluster was then associated with a vector of features. These features were the syntactic features weighed using Inverse Cluster Frequency (ICF). ICF is similar to IDF and assigns higher weights to features appearing in a small number of basic clusters.

Finally in the last stage, the basic clusters are meta-clustered to obtain relations between entities. In particular, given a pair of basic clusters, their entities are retrieved, and all possible parallel mappings of entities from both basic clusters are identified. Each mapping is then assigned a score according to the aggregate similarity of its entity vector pairs. The highest mapping is finally kept, and transformed to an \( n \) by 2 table, where \( n \) is the number of entity pairs (defined in the mapping), and the two columns enumerate the entities of each basic cluster.

Table 5 shows an example of a table extracted by their method, in which we observe that the first column refers to earthquakes and the second refers to locations that were hit. We also observe an incorrect relation which defines a president of the USA as an earthquake. This relation was extracted due to the inherent ambiguity in natural language, e.g. *Bush flew to Texas* as opposed to *Hurricane Katrina flew to New Orleans*. This false positive example also shows that their method does not deal with ambiguous features.
In their evaluation, they examined 48 meta-tables chosen randomly out of 101, which were extracted as a result of clustering their news articles. These 48 tables were characterized as consistent, if at least half of their rows depicted a correct relation instance. 36 out 48 tables were deemed to be consistent (75%). Finally, out of a total 161 rows, 118 (73%) were describing a correct relation between entities.

The Dogma workbench [38,39] is another distributional clustering approach which extracts relations between noun phrases than only relations between entities. In their method they have focused on two types of relations, i.e. hyponymy and meronymy relations.

Initially, Dogma identifies object/verb and subject/verb relations to create a list of basic clusters. Each basic cluster is a set that contains those nouns that appear as subjects or objects of the same verb.

Reinberger and Spyns [38,39] apply three weighting schemes to filter out noisy grammatical relations. The first one is simple frequency, the second is the conditional probability of a noun given the grammatical relation, and the third is the Pointwise Mutual Information between a noun and the grammatical relation.

For example, in the automobile domain Dogma would extract the following clusters of nouns. In this example, there are two clusters, each one containing nouns appearing as objects/subjects of the corresponding verbs.

- **drive_object**: BMW, Audi, Mercedes
- **sell_subject**: BMW, Mercedes, Fiat

Given a list L of basic clusters (as above), three hierarchical clustering algorithms are suggested. The first is a soft clustering algorithm, which allows a word to belong to more than one cluster, taking into account the polysemy of words.

Clusters in L are pairwise compared using three different measures. The first one is the number of common elements between two clusters CE, the second is the number of nouns appearing only in the first cluster N1, and the third is the number of nouns appearing only in the second cluster N2. If these calculated measures respect some empirically predefined values, then the clustering is considered valid. The literature does not provide details about the values of their parameters.

After the initial cluster has been compared to all other classes, all the valid matchings are compared and the one producing the largest new cluster is kept and added to L. Each time a new cluster is created, the 2 clusters are removed from L. In the above example for the classes drive_object, sell_object CE=2, N1=1and N2=1. Thus if these respect some pre-defined values, a new cluster can be created, which contains

<table>
<thead>
<tr>
<th>Earthquake</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rita</td>
<td>Texas</td>
</tr>
<tr>
<td>Katrina</td>
<td>New Orleans</td>
</tr>
<tr>
<td>Wilma</td>
<td>Florida</td>
</tr>
<tr>
<td>Bush</td>
<td>Texas</td>
</tr>
</tbody>
</table>

*Table 5: Meta-table example extracted by the method of Shinyama & Sekine [42]*

Deliverable 4.2 - Public

41/52
the items BMW, Audi, Mercedes, Fiat.

The second suggested algorithm is a hard clustering algorithm that works in a similar fashion as its soft counterpart. However, the input to this algorithm is a list of nouns each one associated to a cluster of verbs. For instance, the above example will be transformed to the following:

- BMW: drive, sell
- Audi: drive
- Mercedes: drive, sell
- Fiat: sell

Nouns are then clustered according to the similarity of clusters of verbs by taking into account the same heuristics as in soft clustering. Classes are compared two by two and if the similarity measures respect some empirically predefined values, then the matching is possible. A third clustering strategy they suggested was to combine the results of both soft and hard clustering techniques, by only keeping the soft clusters, which have a certain degree of similarity with the hard clusters.

Evaluation of a Dogma taxonomy considers the recall/precision of its clusters with respect to word pairs (inside the clusters), which are also related in WordNet with one of the following relations: hyponymy/hypernymy, and meronymy. Their evaluation has shown that the merging of soft and hard clustering provides better results than soft clustering which produces too many clusters, and hard clustering which produces a few clusters only.

4.2 Methods based on hypothesis testing

Ciaramita et al. [10] developed a method for learning semantic relations between concepts within the domain of biology. Relations are learned from the Genia corpus [34] by means of syntactic and statistical analysis of text.

The Genia corpus [34] consists of 18546 sentences and is hand-tagged with biological entities (e.g. DNA). Ciaramita et al. [10] note that they used this hand-tagged corpus in order to learn relations between entities, instead of applying an NER system on the raw text version of Genia. In their setting they also excluded sentences, which included entities that were linguistically realised by complex conjunctions/disjunctions (e.g. erythroid, myeloid and lymphoid cell types). They also excluded sentences with more than 100 words, in effect ending up with a total of 18333 sentences.

These sentences were parsed with a statistical parser [9] by first replacing each entity in a sentence with its corresponding entity tag. For example the entity protein molecule would be replaced in each sentence of the corpus with its corresponding tag Protein_Molecule. This technique was applied in order to avoid cases, in which the parser is unable to connect the components of a complex and long entity.
The parse trees produced that way are much simpler, since they do not split the entities across phrases. For each tree, they generated a dependency graph, in which each word (lemma) is associated with one governor, where a governor is defined to be the syntactic head of the phrase closest to the word. For example, consider the sentence: Androstenedione taker should receive information to have monitored blood levels of hormones and cholesterol, and its corresponding parse tree.

(S (NP Androstenedione_takers) (VP should (VP receive (NP information) (S (VP to (VP have (NP (NP (NP hormones) and (NP cholesterol)))))))))

Let us also assume that noun phrases Androstendione_takers, hormones, and cholesterol are three named entities. The governor of Androstendione_takers is receive, the governor of hormones is the preposition of, while the governor of cholesterol is hormones.

A relation \( r_{ij} \) between entities \( i, j \) is defined as the shortest path in the dependency graph connecting the two entities. Using this approach, Ciaramita et al. identified 172446 paths. For the sake of simplicity, they restricted the selection of paths by only considering those, whose first entity is a subject of a verb \( v \) acting as its governor. In addition, the following constraints were imposed:

- The second entity \( j \) is governed by \( v \) under a verb phrase.
- The second entity \( j \) is governed by \( v \) under a prepositional phrase (PP).
- The second entity \( j \) is governed by \( v \)'s direct/indirect object.
- The second entity \( j \) is governed by a PP modifying the direct/indirect object.

In the next stage, the target is to identify for a pair of entities these relations, which are strongly associated with that pair. To accomplish their task, they apply Dunning’s log-likelihood statistic [16] in a manner similar to collocation extraction. Specifically, let \( A \) be a pair of entities and \( r \) a relation between them. Table 6 shows the observed frequencies of the entity pair and the relation in the corpus, where for example the cell (2,2) denotes the number of co-occurrence of the entity pair under relation \( r \), the cell (2,3) denotes the number of co-occurrence of the entity pair under all relations excluding \( r \), and so on.

<table>
<thead>
<tr>
<th></th>
<th>( A )</th>
<th>(-A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r )</td>
<td>Count(Ar)=213</td>
<td>Count(A-r)=2439</td>
</tr>
<tr>
<td>(-r)</td>
<td>Count(A-r)=23279</td>
<td>Count(-A-r)=240338639</td>
</tr>
</tbody>
</table>

Table 6: Observed frequencies for an Entity pair \((A)\) and a Relation \( r \)
Once the observed table has been constructed, one formulates the hypothesis that \( A \) and \( r \) do not co-occur more frequently than expected at chance, i.e. \( P(A, r) = P(A) \times P(r) \). Using this hypothesis a second table is created (Table 7), that gives the expected values under the model of independence.

<table>
<thead>
<tr>
<th></th>
<th>( A )</th>
<th>( -A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r )</td>
<td>Count(Ar)=2.58</td>
<td>Count(A-r)=2649.4</td>
</tr>
<tr>
<td>(-r)</td>
<td>Count(A-r)=23489.4</td>
<td>Count(-A-r)=2.403e7</td>
</tr>
</tbody>
</table>

*Table 7: Expected frequencies for an Entity pair \( (A) \) and a Relation \( r \) under independence*

The counts in the two tables are compared using the log-likelihood statistic (Equation 21), where \( o_{ij} \) refers to the observed frequencies and \( e_{ij} \) refers to the expected frequencies under independence. In this example, the value of the statistic is 1476.92. As \( G^2 \) is chi-squared distributed, one can use the chi-square tables to reject or accept the hypothesis at a given confidence level (in their setting \( \alpha =0.005 \)).

\[
G^2 = 2 \sum_{i,j} o_{ij} \log \frac{o_{ij}}{e_{ij}}
\]

*Equation 21: Log-likelihood formula*

In the last stage of their method they mention that while the previous stage can extract the significant relations between two entities, it does not however lead to a generalization of relations. For example, one might find several significant relation instances between many different kinds of protein and DNA. This can be evidence that the relation applies to all types of protein and DNA.

To generalize their relations they use the algorithm of Clark & Weir [12] in combination with the Genia hierarchy. The generalization algorithm takes as input a relation \( r \), a class \( c \) and a syntactic slot \( s \) and outputs a new class \( c' \), which is either \( c \) or one of its ancestors in the hierarchy depending on their generalization strength. This accomplished by checking through a chi-squared test whether the probability \( p(r \mid c, s) \) is significantly higher than the probability \( p(r \mid c', s) \), where \( c' \) is an ancestor of \( c \). If this is false, then \( c' \) provides a good generalization. The process iterates by moving upwards in the hierarchy until a significant difference is found. Note that this process was applied for each relation and for each entity (class) in a relation, while the log-likelihood statistic was again used to determine the significance of changes.

The evaluation of the first stage of their method considered the percentage of relations, which were correct (precision). They report a 76.5% precision, i.e. 150 out of the 196 relations were correct and 46 incorrect. They identified that almost half of their errors is caused by incorrect links in the dependency graphs, and specifically by the inability of their method to handle noun co-ordination and produce the correct dependencies. The remaining errors were caused by data sparsity issues, which causes the log-likelihood statistic to overestimate events and fail.
Evaluation of the generalization method showed that out of 113 generalized patterns 60 (53.1%) were correct and the rest incorrect. The relatively low performance is mainly caused by the fact that the Genia ontology is not an IS-A hierarchy.

### 4.3 Summary

Unsupervised methods relax all the supervision requirements of supervised and weakly-supervised methods by extracting relations without the use of any apriori knowledge or hand-tagged data. As a result, they have the ability to be applied to different domains and text genre.

Typically, unsupervised approaches formulate the problem of relation extraction as a clustering problem, in which the target is to cluster pairs of entities that share the same semantic relation. A rather different approach has also been described in section 4.2, which is based on hypothesis testing.

In general, unsupervised approaches have a worse performance than supervised or weakly supervised methods. Although, to the best of our knowledge, there is no study in relation extraction showing this, evaluations on similar NLP tasks show that supervised and weakly-supervised approaches have superior performance.

A further drawback of current relation extraction clustering techniques is the labelling of extracted clusters. The clusters generated by such a RE system need to be labelled in such a way as to denote the relation of interest, i.e. the relation that holds between the entity pairs in the cluster. The methods described above have disregarded this problem, and evaluated directly the entity pairs within the clusters. In a real-world application however, one needs to assign cluster labels in effect imposing a type of supervision.

Additionally, the evaluation of the RE approaches ignores the significance of the clustering algorithm. For example, it is unclear whether flat clustering can perform better than traditional HAC, whether divisive HAC performs better than bottom-up HAC as several studies suggest [31], and more importantly what the optimum number of clusters (level at the binary tree of HAC) would be to group the entity pairs. The estimation of the appropriate number of clusters is an open research issue in the community and in most cases is dealt by the use of parameters which control the fine-grainedness of a clustering solution [51]. As a result, the use of parameters might impose another supervision factor within these clustering approaches.
Conclusions

This report has provided a detailed review of the current-state-of-the-art in relation extraction. Specifically, RE approaches were divided into three categories and for each one of them a number of different methods were presented and discussed. Our critical survey has provided the following important factors that will guide us in the next stages of the project.

- Amount of supervision
- Feature selection
- Machine learning algorithm

It is straightforward from the previous sections that supervised methods are not portable to different domains or applications due to the lack of adequate training data. On the other hand, unsupervised methods have the ability to overcome the knowledge acquisition bottleneck. However, their performance in most NLP tasks is worse than the performance of supervised algorithms.

Note however that this does not mean that supervised methods such CRFs or SVMs cannot be used. In contrast, these methods can still be useful, if we are able to cast their operation in a semi-supervised or unsupervised setting. Specifically, if generate training data automatically then we will be able to exploit SVMs or CRFs. In fact, the high precision of statistical and pattern-based approaches can be a useful source for creating annotated data automatically.

For feature selection, most approaches either adopt a bag-of-words model or exploit syntactic dependencies. The use of a shallow parser seems to be a better choice for generating dependencies due to its tolerance to ungrammatical sentences. This is especially useful for data gathered from the web.

More importantly, the evaluation of different type of kernels [13] shows that the structural information contained in syntactic dependencies provides information extremely useful for relation extraction. Despite this, the combination of a bag-of-words and syntactic dependency kernel has the highest performance. This shows that co-occurring information captures knowledge not encoded in parse trees. Overall the inclusion of both types of features seems to be the most appropriate choice.

Finally, the choice of the machine learning algorithm is of equal importance to the above. We have seen that discriminative models generally have a better performance than generative ones, mainly due to the fact that they are directly optimised to model the conditional distribution of two random variables. CRFs and SVMs are the standard choices in supervised techniques while it is not clear whether one performs better than the other.
Weakly-supervised techniques use statistical formulas or statistical models in order to filter out noisy patterns or relations. In most cases, the formulas used are quite intuitive, yet ad-hoc and unprincipled. In contrast, the use of statistical models allows a clear interpretation of the association between entities in a relation.

In unsupervised methods, the choice of the clustering algorithm affects the granularity of the clustering solution, hence the performance of relation extraction. Current work in RE lacks evaluation of different clustering methods in this task.

Additionally, current work in RE has not employed graph-based clustering widely used in related tasks (information retrieval, word sense induction) with good results [3, 25, 49].
References


## Document Updates

<table>
<thead>
<tr>
<th>Version⁢¹</th>
<th>Date⁣</th>
<th>Updates and Revision History⁣</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>20090925</td>
<td>25/09/2009</td>
<td>Introduction write-up</td>
<td>Ioannis Klapaftis</td>
</tr>
<tr>
<td>20090926</td>
<td>26/09/2009</td>
<td>Supervised approaches write-up</td>
<td>Ioannis Klapaftis</td>
</tr>
<tr>
<td>20091001</td>
<td>01/10/2009</td>
<td>Unsupervised approaches write-up</td>
<td>Ioannis Klapaftis</td>
</tr>
<tr>
<td>20091004</td>
<td>04/10/2009</td>
<td>Weakly supervised approaches write-up</td>
<td>Ioannis Klapaftis</td>
</tr>
<tr>
<td>20091007</td>
<td>07/10/2009</td>
<td>Conclusions</td>
<td>Ioannis Klapaftis</td>
</tr>
<tr>
<td>20091012</td>
<td>12/10/2009</td>
<td>Stylistic changes (Mikołaj Leszczuk remarks) - released for internal review</td>
<td>Ioannis Klapaftis</td>
</tr>
<tr>
<td>20091018</td>
<td>18/10/2009</td>
<td>Added more examples</td>
<td>Ioannis Klapaftis</td>
</tr>
<tr>
<td>20091030</td>
<td>30/10/2009</td>
<td>Spelling and stylistic changes</td>
<td>Ioannis Klapaftis</td>
</tr>
</tbody>
</table>

---

³ In form of “vYYYYMMDD”; Version number and edition should correspond to the actual document name conventions.

⁴ In form of “DD/MM/YYYY”

⁵ Attach as appendix document reviews when appropriate; describe also the current status of the document e.g. “released for internal review”, “released for comments from partners”