Information Extraction with RapidMiner

Felix Jungermann


Abstract. In this paper we present the Information Extraction (IE)-
plugin for the open source Data Mining (DM) software RapidMiner\(^1\) [Mierswa et al., 2006]. The IE-plugin can be seen as an interface between
natural language and IE- or DM-methods, because it converts docu-
ments containing natural language texts into machine-readable form in
order to extract interesting information like special entities and relations
between those. The plugin is very modular and easy to use, which makes
it more applicable for different domains and tasks.

Key words: Natural Language Processing, Information Extraction, In-
formation Extraction System, Named Entity Recognition, Relation Ex-
traction, Structured Methods

1 Introduction

Nowadays more and more information is available spread all over the internet.
The information is present on websites – containing pure text on the one hand
and html-code on the other hand –, in documents (pdf-documents for instance),
or in log-files and so on. To process this daily growing huge amount of information
manually is impossible.
Therefore IE-techniques are used for the automatic identification of selected types
of entities, relations, or events in free text, as [Grishman, 2003] says.
While some IE-systems process IE-tasks like for instance Named Entity Recog-
nition (NER) in a somehow black-boxed way, we present a very modular system,
which can easily be adjusted and extended for already known or new tasks.
In the following section we give a short overview of the history of IE. In section 3
we present the state-of-the-art DM software RapidMiner. In section 4 the special
aspects of natural language processing and how they are respected in RapidMiner
will be explained. Some principles concerning automatic IE are defined in section
5. The IE-plugin and an exemplary text-handling task will be presented in section
6. Finally section 7 will give a conclusion about current efforts and future work.

2 Information Extraction

Beside the definition of [Grishman, 2003], [Cardie, 1997] defines IE as a kind of
summarization of texts according to some (predefined) topic or domain. Both

\(^{1}\) http://www.Rapid-I.com
definitions aim at a deeper text understanding.
But before one gets to know how to gain more knowledge from unknown texts,
one should have a look at the history of IE.

The field of IE was born together with the Message Understanding Conference (MUC) [Grishman and Sundheim, 1996] which were scientific, shared tasks, offering texts of a special domain. Participants in these tasks had to answer specific questions concerning several special mentions in the text. One of the first domains for example was 'terrorist acts'. The questions to answer were 'Who was the actor?', 'Who was the victim?', 'What happened?', and so on.
Answering these questions automatically is a non-trivial task. Because of that, many systems were handcrafted and contained a huge amount of manually designed rules, resulting in systems biased towards a specific domain.
Using these systems for other domains was impossible or very unattractive, because much human effort had to be invested to tune all the rules manually.
The understanding of the semantic of texts was not very deep, and therefore tasks were defined that need to be processed in every domain for a deeper understanding: namely co-reference analysis, word sense disambiguation and predicate-argument structure.
Especially a new task of the sixth MUC – namely NER – got very popular. In its original form, the task was to extract predefined (very basic) named entities like person names, locations and organizations.
The goal to create a new task was successful, but NER is in no way domain-independent, because every special domain needs special entities to be extracted.
A very popular application area of NER is the biological domain which uses NER for identifying genes, proteins and so on. That task is called 'bio-entity recognition', like in the BioNLP of 2004 ([Kim et al., 2004]).

Those entities are the first semantic milestones on the way to a better, automatic text understanding.
But, effective – domain- and language-independent – systems should not be based on handcrafted linguistic rules. Although manually tuned systems are sometimes better in terms of precision and recall (see section 5.3), machine-learning or statistical methods are used, because they are more flexible and do not rely on heavy human efforts.
Machine-learning systems used for NER are hidden Markov models [Rabiner, 1989], maximum entropy Markov models [McCallum et al., 2000], conditional random fields [Lafferty et al., 2001] and structured support vector machines [Tschantaridis et al., 2005].

3 RapidMiner

In this section we will give a short overview about the concepts of traditional DM. The opensource DM-software RapidMiner – one of the best DM tools ² –
is presented as an exemplary environment for DM-tasks.

One can say that Knowledge Discovery in Databases (KDD) is the whole field of research of extracting new and interesting knowledge out of (masses of) information, which in addition might be very unstructured. The term KDD is often used equivalently in one context with the term of DM, which describes the techniques used to extract that knowledge. Because the amount of information is often huge, the techniques must be scalable to work efficiently.

Traditionally this extraction was done on databases containing related entities. These relations are decomposed and put into a flat form in order just to have one table to learn from.

As it is the same convention in RapidMiner from now on this table is called exampleset, and this exampleset, again, contains examples. In addition, examples consist of attributes. The special label-attribute is the one to predict from unlabeled new examples. Table 1 shows such an exampleset. In this exampleset the label is a binary one.

<table>
<thead>
<tr>
<th>Example</th>
<th>Attribute_1</th>
<th>Attribute_2</th>
<th>...</th>
<th>Attribute_m</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>example_{n-1}</td>
<td>0.1</td>
<td>true</td>
<td>...</td>
<td>...</td>
<td>'B'</td>
</tr>
<tr>
<td>example_n</td>
<td>0.2</td>
<td>false</td>
<td>...</td>
<td>...</td>
<td>'Re'</td>
</tr>
<tr>
<td>example_{n+1}</td>
<td>0.2</td>
<td>true</td>
<td>...</td>
<td>...</td>
<td>'Bc'</td>
</tr>
<tr>
<td>example_{n+2}</td>
<td>0.05</td>
<td>true</td>
<td>...</td>
<td>...</td>
<td>'BB'</td>
</tr>
</tbody>
</table>

### 3.1 Process-view

The process-view (see Figure 1) of RapidMiner offers a modular and pipelined view on the processed experiment. The process normally consists of four stages:

- the input-stage,
- the preprocessing-stage,
- the learning-stage,
- and the evaluation-stage.

These stages correspond to the CRISP-model ([Chapman et al., 1999]) which is a well-known standard for DM processes.

### 4 Information Extraction aspects

Remembering section 2, one sees that IE can be seen as a special case of traditional data mining.
Examples will contain the tokens of the text, whereas the whole document-collection can be seen as the example set. The attributes are syntactic or semantic pieces of the tokens, and the label is the special semantic information one likes to predict.

One of the most important aspects in IE is the sequential character of texts. For the categorization of documents it is often sufficient to just use the so called bag of words, which is a simple index list of the contained words, to classify an unknown document – having seen enough classified documents during the training phase. But for extracting relevant information, IE needs the words itself and its contexts, e.g. to predict if it is an entity or not.

[McDonald, 1996] describes external and internal evidence as necessary to extract the correct semantics for a given word. The fact that RapidMiner is a DM software is remarkable because in ‘traditional’ DM a special example (and its attributes) originally is independent of the other examples – although they might likely share patterns. For IE all documents have to be converted to proper examples first. During conversion it is necessary to maintain the sequential character of documents and especially of sentences. Standard ML operators like sampling have to be adapted to meet the needs of the plugin in order to not destroy the structural information of the language constructs.

One way to maintain such structural characteristics is to associate words and sentences with identifiers, usually by using counter variables. Table 2 shows a possible example set for IE-tasks. For a better understanding the reader should have a look at section 6 which shows an example process for IE.
Table 2. Possible IE-exampleset in RapidMiner

<table>
<thead>
<tr>
<th>Token</th>
<th>Doc.No.</th>
<th>Sent.No.</th>
<th>TokenNo.</th>
<th>Attribute1</th>
<th>Attribute2</th>
<th>...</th>
<th>Attribute_m</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>goes</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>'g'</td>
<td>'s'</td>
<td>...</td>
<td>...</td>
<td>aaaa</td>
</tr>
<tr>
<td>to</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>'t'</td>
<td>'o'</td>
<td>...</td>
<td>...</td>
<td>aa</td>
</tr>
<tr>
<td>Hamburg</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>'H'</td>
<td>'g'</td>
<td>...</td>
<td>...</td>
<td>Aaaaaa</td>
</tr>
<tr>
<td>because</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>'b'</td>
<td>'e'</td>
<td>...</td>
<td>...</td>
<td>aaaaaa</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

5 Automatic-Information Extraction

In this section we explain two of the most important tasks in IE and their techniques that are present in the IE-plugin.

5.1 Named Entity Recognition (NER)

Traditionally, NER is the task to predict the best label-sequence \( Y \) for a given observation-sequence \( X \), having seen a sufficient amount of training-pairs

\[
<x^{(i)}; y^{(i)}>
\]

There are some unsupervised approaches for NER (which do not need masses of training-data) ([Nadeau et al., 2006], [Collins and Singer, 1999], [Etzioni et al., 2005]) but they heavily rely on seed-examples and knowledge about the wanted Named Entity (NE)s.

The current state-of-the-art technique for NER are CRF we will explain now in more detail.

Conditional Random Fields (CRF) CRFs are undirected graphical models and have first been proposed by [Lafferty et al., 2001].

Definition 1. Let \( G = (V,E) \) be a graph such that \( Y = (Y_v)_{v \in V} \), so that \( Y \) is indexed by the vertices of \( G \). Then (\( X,Y \)) is a CRF, when conditioned on \( X \), the random variables \( Y_v \) obey the Markov property with respect to the graph:

\[
p(Y_v|X,Y_w, w \neq v) = p(Y_v|X,Y_w, w \sim v)
\]

where \( w \sim v \) means that \( w \) and \( v \) are neighbors in \( G \).

The generated label-sequence is calculated on the whole graph conditioned on the complete observation-sequence. The probability of the graph is defined via two kinds of features:

- transition-features: \( f(e,y|e,x) \) defined on the edges of the graph and
- state-features: \( g(v,y|v,x) \) defined on the vertices.
In the simplest and most important example for modeling sequences, $G$ is a simple chain or line:

$$G = (V = \{1, 2, ..., m\}, E = \{(i, i + 1)\})$$

If the graph is a tree one can write the conditional distribution of $Y$ and $X$ as follows:

$$p_\theta(y|x) \propto \exp \left( \sum_{e \in E, k} \{\lambda_k f_k(e, y|e, x)\} + \sum_{v \in V, k} \{\mu_k g_k(v, y|v, x)\} \right)$$

Using the log-likelihood we can derive the following optimization function $L(\lambda)$ as:

$$L(\lambda) = \sum_i \left( \log \frac{1}{Z(x^{(i)})} + \sum_k \lambda_k F_k(y^{(i)}, x^{(i)}) \right)$$

The goal is to optimize $\lambda$ which is the weight-vector for the CRF-features such that $L(\lambda)$ is maximized. This optimization can be done by using quasi-newton optimization (e.g. L-BFGS, [Liu and Nocedal, 1989]).

5.2 Relation Extraction (RE)

If the task of finding the entities is sufficiently done, one can look for relations between these entities. The scientific field of finding relations between entities has become popular since the ACE-tasks\(^3\). Especially the relation detection and classification task in 2004 [LDC, 2004] has heavily been worked on. If all entities in a sentence have been found, every possible pair of two entities is combined to a relation-candidate in order to find out wether there is a relation and to predict the corresponding relation type.

**Composite Kernel** The state-of-the-art techniques in this field of research are treekernel-methods based on the research done by [Haussler, 1999] and [Collins and Duffy, 2001]. A recent extension is the combination of treekernels on the one hand with linear kernels on the other hand, resulting in a so called composite kernel. The composite kernels have shown to achieve better results than with just one of these kernels ([Zhang et al., 2006], [Zhang et al., 2007]). These techniques are heavily based on structural information (parse-trees). Therefore the flat data-structure of RapidMiner is not well-suited for this task. Nevertheless treekernels (and composite kernels) are available in the IE-plugin. A treekernel proposed by [Collins and Duffy, 2001] is an efficient way to compare

\(^3\)http://projects.ldc.upenn.edu/ace/
two trees:

\[
K(T_1, T_2) = \sum_i \left( \sum_{n_1 \in N_1} I_{\text{subtree}_i}(n_1) \right) \left( \sum_{n_2 \in N_2} I_{\text{subtree}_i}(n_2) \right)
= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \Delta(n_1, n_2)
\]

where \(T_1\) and \(T_2\) are trees and \(I_{\text{subtree}_i}(n)\) is an indicator-function that returns 1 if the root of subtree \(i\) is at node \(n\). \(\Delta(n_1, n_2)\) represents the number of common subtrees at node \(n_1\) and \(n_2\). The number of common subtrees finally represents a syntactic similarity measure and can be calculated recursively starting at the leaf nodes in \(O(|N_1| |N_2|)\).

The composite kernel combines a treekernel and a linear kernel. [Zhang et al., 2006] therefore present linear combination for instance:

\[
K_1(R_1, R_2) = \alpha K_L(R_1, R_2) + (1 - \alpha) K(T_1, T_2)
\]

where \(R_1\) and \(R_2\) are relation-candidates. Tuning the \(\alpha\)-value changes the influence of the specific kernels.

![Fig. 2. Schema of the IE-plugin processing](image)

5.3 Evaluation measures

For the evaluation of results for IE-tasks, precision, recall and f-measure are widely used. They are defined as follows:

\[
\text{precision} = \frac{|\text{correctly labeled tokens}|}{|\text{tokens}|}
\]
\[
\text{recall} = \frac{|\text{correctly labeled tokens}|}{|\text{labeled tokens}|}
\]
\[
\text{f-measure}_n = (1 + n) \frac{\text{precision} \times \text{recall}}{n \times \text{precision} + \text{recall}}
\]
A special label is reserved for tokens which are `not labeled`. Labeled tokens have another label but not this special one.

6 Plugin

The IE-task in the plugin is following a simple scheme (Figure 2). The major steps are described in detail in the following part.

**Text input and tokenization** Texts or documents are available in various forms like plain ascii-files, xml- or html-files and pdf-files for example. Some text-formats are easy to process and some need additional parsing effort. The plugin offers an input-operator which can handle ascii-, html- and pdf-files. After loading a document, it is present as as a continuous block of text in one table cell, which is not accessible in a profitable way. As a first step tokenizers have to be used to split up the text. Up to now, a trivial sentence- and word-tokenizer is available. Figure 3 shows a html-file in RapidMiner after being loaded and tokenized in sentences. While tokenizing splits the texts into smaller parts the original structure is still kept available. So, if one wants to process documents on the word-level, the sentence-level (the level above) is still present (see Figure 5) and can be used for processing the features. These features basically can be extracted from the current position of the document (sentence, word, ...) and from circumfluent positions (see 6).

**Annotation and visualization** A visualization operator allows to view the documents and to annotate parts of them. One can select attributes which shall be visualized, in order to highlight different aspects of the text. Figure 4 shows a document with some annotated words.

**Preprocessing** The most important part of the IE-plugin is the preprocessing. The preprocessing-operators are used to enrich the document and its tokens with syntactical information which will later be used to extract semantic information. Tokens can be enriched with contextual information as well as they can deliver inner-token information. One should keep in mind that different tasks need different preprocessing. To use machine learning algorithms for IE, one has to enrich the documents, sentences or words with attributes extracted from themselves (internal evidence) and from their context (external evidence). The plugin offers preprocessing-operators which can be used in a very modular way. There are operators working on document-, sentence- and word-level. Figure 5 shows a document after tokenization and preprocessing. Every word of the document is represented by one row (example) in the exampleset, so the whole document is contained in the exampleset. The columns represent the attributes for each example (word). Here, in addition to the word itself (word\_0), the prefix of the current word of length 3 (prefix\_0,3) is used as an attribute.
Fig. 3. The html-file after being loaded and tokenized in RapidMiner

Fig. 4. The html-file after being tokenized in words and annotated
Learning The IE-plugin offers operators for NER and for RE. RapidMiner learners deliver so called models which equate to a function and can be applied to an (unlabeled) exampleset. Until now, for NER, the Conditional Random Fields (CRF) operator (see section 5.1) can be used. For RE the treerkernel operator (see section 5.2) should be used. The underlying techniques have been described before. The implementation of these learning algorithms has been kept modular to allow the combination of various methods. Due to this modularization the CRF-learner can be combined with various optimization-methods such as quasi-newton-methods or evolutionary algorithms (particle swarm optimization for example).

DM-usage The calculated model can – of course – be used for the extraction of interesting information from unseen documents, but after having processed entities or relations or every other information one can easily build up a new exampleset containing the extracted information to gain additional knowledge.

7 Conclusion and Future Work

We presented the IE-plugin\(^4\) for the opensource datamining software RapidMiner. The IE-plugin enriches the functionalities of RapidMiner with IE-related techniques.

The flat internal database-like data-format has the advantage of easily allowing flat and simple syntactical features to be added or removed. A very broad and

\(^4\) the plugin should be available at [http://rapid-i.com/content/view/55/85/](http://rapid-i.com/content/view/55/85/)
deep access to the data and the learners is possible. Because of the modular architecture it is possible to change many settings – in contrast to black-boxed systems.

A minor disadvantage of the current implementation is, that structured information cannot be handled in an easy and adequate way, yet. Therefore the next steps will be to develop new internal datatypes to handle structural information in a better way. Additionally, to evaluate the performance of the implemented IE operators currently tests are running on well known datasets for RE to compare the results of the plugin with other systems.

References


Abbreviations

CRF Conditional Random Fields
DM Data Mining
IE Information Extraction
KDD Knowledge Discovery in Databases
ML Machine Learning
MUC Message Understanding Conference
NE Named Entity
NER Named Entity Recognition
RE Relation Extraction