



Methodology and Architecture for Information Extraction

Distribution: Project internal

MedIEQ

Quality Labelling of Medical Web content using
Multilingual Information Extraction

National Centre for Scientific Research "Demokritos"
Teknillinen Korkeakoulu – Helsinki University of Technology
Universidad Nacional de Educacion a Distancia
Col.legi Oficial de Metges de Barcelona
Zentralstelle der deutschen Ärzteschaft zur Qualitätssicherung in der
Medizin
Vysoká Škola Ekonomická v Praze
I-Sieve Technologies Ltd.

2005107 **D8**

December 2006

Project ref. no.	<i>2005107</i>
Project acronym	<i>MedIEQ</i>
Project full title	<i>Quality Labelling of Medical Web content using Multilingual Information Extraction</i>

Security (distribution level)	<i>Project internal</i>
Contractual date of delivery	<i>30 September 2006</i>
Actual date of delivery	<i>22 December 2006</i>
Deliverable number	<i>D8</i>
Deliverable name	<i>Methodology and Architecture for Information Extraction</i>
Type	<i>Report</i>
Status & version	<i>Final</i>
Number of pages	<i>37</i>
WP contributing to the deliverable	<i>WP6</i>
WP / Task responsible	UEP
Other contributors	NCSR, UNED, HUT, WMA
Author(s)	M. Labsky, M. Nekvasil, V. Svatek (UEP), V. Metsis, K. Stamatakis, V. Karkaletsis, (NCSR), M. Pöllä, T. Honkela (TKK), F. L. Ostenero, V. Peinado (UNED)
EC Project Officers	Artur Furtado
Keywords	Information Extraction, named entity extraction, instance extraction, machine learning, extraction ontologies
Abstract (for dissemination)	This document proposes the methodology and architecture for the information extraction in MedIEQ. The described information extraction toolkit (IET) consists of multiple components of different types. The information extraction tools process documents and add to these documents simple or structured annotations.

Table of contents

Executive summary	5
1. Introduction.....	6
2. Related work	7
2.1 Named Entity extraction	7
2.2 Instance extraction	10
2.3 Information integration.....	11
2.4 Information extraction tools	11
3. The information extraction methodology	13
3.1 Hierarchy of tools and components.....	13
3.2 Tools and components quick reference.....	13
4. The Information Extraction Architecture	19
4.1 Manager Components APIs.....	19
4.2 Data Container Classes.....	20
4.3 Data Processing Components.....	21
5. Users, user interfaces and use cases	22
5.1 Users & UIs.....	22
5.2 Use cases	22
6. Concluding remarks	26
Bibliography	27
APPENDIX A. Terminology	29
APPENDIX B. Initial set of labelling criteria	30
APPENDIX C. Extraction tool survey	35

Executive summary

MedIEQ is an on-going EC-funded project aiming to:

- Provide a common vocabulary and machine readable schema for the quality labelling of health-related web content and develop tools supporting the creation, maintenance and access of labelling data according to this schema;
- Specify a methodology for the content analysis of health web sites according to the MedIEQ schema and develop the tools that will implement this methodology;
- Integrate these technologies into a prototype labelling system in seven (7) languages (ES, CA, DE, EN, GR, CZ, FI) aiming to assist the labelling experts.

At the time of writing this report, the 1st version of the MedIEQ schema has been finalised. This schema will allow the creation of machine readable content labels.

Work is now underway to develop applications to make use of such labels (generation, maintenance, validation against the MedIEQ schema).

Before that, other applications have to analyze the content of health websites and extract information related to the labelling criteria included in the MedIEQ schema. Two separate toolkits, handling the different levels of content analysis, have been scheduled: the Web Content Collection toolkit (WCC) and the Information Extraction toolkit (IET).

This report focuses on the Information Extraction toolkit (IET).

1. Introduction

The prototype MedIEQ labelling assisting system (also called AQUA, from Assisting Quality Assessment) consists of 5 subsystems or toolkits:

1. the label management toolkit (LAM),
2. the web content collection toolkit (WCC),
3. the information extraction toolkit (IET),
4. the multilingual resources management toolkit (MRM),
5. the monitor-update-alert toolkit (MUA).

LAM manages (generates/validates/modifies/compares) quality labels based on the schema proposed by MedIEQ (an RDF-CL implementation).

WCC identifies, classifies and collects on-line content relative to a number of machine readable quality criteria (proposed by the participating labelling agencies) in seven languages (EN, ES, DE, CA, GR, FI, CZ).

IET analyses the web content collected by WCC and extracts attributes and structured objects for MedIEQ-compatible content labels.

MRM gives access to health-related multilingual resources (like MeSH, ICD or whatever being available); input from such resources is needed in specific parts of both the WCC and IET toolkits.

All data necessary to the different subsystems as well as to the overall AQUA system are stored in the MedIEQ repository.

Finally, MUA handles a few auxiliary but important jobs, including the configuration of monitoring tasks, the MedIEQ repository's entries updates and the alerts to labelling experts when important changes occur in existing quality labels.

This deliverable focuses on the Information Extraction toolkit. IET employs a set of components which will be responsible for the extraction of the parts of information found in each document and the integration of these parts in a set of useful instances. IET is being designed as a generic information extraction toolkit which should make it easy to incorporate any changes or additions to the utilised labelling schemes. In this way, IET could also be used for IE using third-party labelling schemes and within different domains. The IET components will implement documented APIs to ensure they can be invoked by components from other toolkits or even from outside the MEDIEQ project. In addition, some IET components will provide user interfaces for administrators and/or labelling experts. The user interfaces will allow both for running the IET within the MedIEQ system and for using it independently.

Related work on information extraction (IE) is described in section 2. The IE methodology, proposed by MedIEQ, with short descriptions of the components involved, is given in section 3. The proposed IE architecture is described in section 4. Use cases are described in section 5 and section 6 presents our concluding remarks. There are also appendices for the terminology used, the methods that can be used to extract information relevant to the 1st set of the labelling criteria (these criteria are described in section 3 of Deliverable D4.1), and a brief survey on information extraction tools.

2. Related work

In this section we describe state-of-the-art IE methods used to extract information from free-form and structured text. According to the nature of extracted information, we may distinguish between:

- named entity extraction,
- instance extraction.

Named entity extraction aims at identifying values of independent attributes in text (e.g. proper names, dates, numbers). *Instance extraction* tries to extract groups of related attribute values from text. In MedIEQ, some labels, even in the initial set of criteria, will require instance extraction techniques (e.g. person name, address, contact).

According to the type of texts processed, we may identify:

- extraction from free-text,
- extraction from structured (formatted) text.

Extraction from free-text often uses natural language processing (NLP) techniques such as syntactic parsing [22], while extraction from structured texts often relies on visual formatting clues (NLP plays less critical role as structured text is often ungrammatical). In both cases, the extraction features used typically include the words constituting the value and context of the extracted attribute. Additional features often involve word types (e.g. part-of-speech tags) or the presence of certain character-level or word-level patterns.

In MedIEQ, we will be mostly extracting from web sites which present text in a structured manner, therefore we will also try to utilise formatting information if present. Initially we will be able to extract from HTML and text documents, and as the IE toolkit evolves we will add support for other formats such as PDF and MS Office/Open Office. With respect to the nature of the initial label set, we initially do not plan to use NLP techniques other than lemmatizing and part-of-speech tagging (see e.g. [28]). As the label set grows on complexity, we may incorporate more sophisticated techniques such as syntactic parsing.

In the following subsections we first list several state-of-the-art IE methods, which are in most cases used to extract named entities. Selected instance extraction methods are then presented which often build on the results of named entity extraction. Information integration methods are then discussed. Their task is to normalise and compare the extracted labels. Note that the named entity extraction, instance extraction and information integration areas have significant overlaps.

2.1 Named Entity extraction

There are three major (overlapping) families of IE methods: dictionary-based, rule-based and statistical.

A. Dictionary and pattern-based methods

IE can be done by matching parts of documents against dictionaries of known attribute values or against sets of manually defined patterns for the content and context of extracted terms. The advantages of this approach are its simplicity and that there is no need for annotated training data. The disadvantages include coverage of used dictionaries, ambiguities of dictionary terms, problematic manageability of large sets of patterns and problems in combining several patterns together.

To ease the IE process, many rule-based and probabilistic machine-learning methods have been developed. The advantage of machine-learning methods is that they do not rely on manually specified knowledge, which may often be misleading and suboptimal. Instead these methods induce "their own knowledge" from training data. The training data are created by human annotators either manually or semi-automatically and can be reused across different IE methods. A number of annotation tools are available to assist human annotators in creating training data (e.g. the annotation tools of the Ellogon¹ language engineering platform that will also be exploited in the context of MedIEQ). The approach of active learning [4], [10] integrates IE algorithms directly into annotation tools and guides the annotator to label the most needed patterns.

B. Rule induction methods

These are approaches that treat the IE task as learning (inducing) generalised extraction rules from a set of training example labels to be extracted. Particular form of the learned rules depends on the method used. In the following we mention three examples of rule-based methods: (LP)², Rapier and IE by double classification.

(LP)² is an algorithm for adaptive IE from text that induces symbolic rules by learning from a corpus of manually annotated documents [4]. Induction is performed by bottom-up generalisation of examples in the training corpus. Training is performed in two steps: initially a set of tagging rules is learned; then additional rules are induced to correct mistakes and imprecision in tagging. Shallow NLP is used to generalise rules beyond the flat word structure. Generalisation allows for a better coverage on unseen texts, as it limits data sparseness and over-fitting in the training phase.

The RAPIER algorithm [3] uses pairs of sample documents and filled templates to induce pattern-match rules that directly extract fillers for the slots in the template. RAPIER is a bottom-up learning algorithm that incorporates techniques from several inductive logic programming systems. The induced patterns may include constraints on the words, part-of-speech tags and semantic classes present in the extracted item and in the surrounding text.

¹ <http://www.ellogon.org>

IE via double classification is one example of approaches to IE which build upon classification of phrases, words or gaps between them. In one such approach [30], each document is first divided into sentences. For each sentence a Naive Bayes classifier working on a bag-of-words representation of the sentence is used to decide whether it is relevant or not. By doing this, a set of relevant sentences s_1, s_2, \dots per document ordered by a measure of certainty is obtained. In a second step this approach uses a rule-based classifier that decides for each word in the sentence whether it belongs to the entity to be extracted.

C. Probabilistic models

Many problems in natural language processing can be formulated as statistical classification problems, in which the task is to estimate the probability of some event (e.g. some phrase being the value of an extracted attribute) occurring in an observed context (e.g. the phrase's content, surrounding words and formatting). The learning process thus involves estimation of different types of probability distributions, which are then used for extraction. One disadvantage lies in the need for typically large amounts of training data. Underestimation of training data size leads to data sparseness problems and to higher error rates.

Hidden Markov Models (HMM) have strong statistical foundations and are computationally efficient to learn and evaluate due to the existence of established learning and search algorithms based on dynamic programming [26]. HMM training consists in estimating a generative probability distribution that best suits the training data. HMMs proved to be well suited for natural language processing tasks such as speech recognition or part-of-speech tagging. They also have been successfully applied to many sub-domains of information extraction: the named entity extraction task; to the task of recovering a sequence of entities occurring in close proximity (dense extraction); as well as the sparse extraction task, in which the aim is to extract relevant phrases from documents containing much irrelevant text [12], [1], [18].

There are several disadvantages to using HMMs. One major disadvantage lies in the fundamental independence assumptions of Markov Models, which make it hard to incorporate non-independent features (e.g. word membership in two related domain dictionaries). Second, as any statistical technique, HMMs require relatively large amount of training data. Another problem when applying HMMs to IE is the selection of an appropriate state-transition structure (topology) of the model. In many cases the topologies are designed manually but there exist hill-climbing approaches, which are able to find an appropriate topology given training data [12].

Similarly to HMMs, Maximum Entropy Models (MEMs) are trained from large corpora, but the type of the estimated probability distributions is conditional instead of generative. MEMs make it easier to incorporate large numbers of correlated features. The distribution is trained according to the principle of maximum entropy, which states that the best probability distribution is the one that maximises the uncertainty given the constraints known to the experimenter. The constraints are typically

determined by feature counts observed in training data. The distribution is trained using iterative algorithms such as the Generalised Iterative Scaling algorithm. The algorithms used for the extraction itself resemble in many aspects the HMM extraction algorithms; further see [23].

Conditional Random Fields (CRFs) is another statistical technique, which aims to bring together advantages of generative models (such as HMMs or stochastic grammars) and conditional models (such as MEMs). Like conditional models, CRFs handle well multiple correlated features, and like generative models, they retain the ability to trade-off local extraction decisions with global ones. The disadvantages include computation-intensive iterative training procedure and currently small number of existing CRF toolkits. For further information see [20] and [29].

Of course the above list is not exhaustive; important methods not mentioned include classification algorithms such as the Perceptron [6] and Support Vector Machines [31]. There are a large number of other trainable machine learning algorithms described e.g. in [24] and implemented in as part of open-source toolkits such as the Weka system [32].

2.2 Instance extraction

Instance extraction methods are closely related to the previously described IE methods since they often come right after identifying the candidates for the individual labels. The task of instance extraction is to find out which label candidates belong together; i.e. which labels form an instance that can be stored in a database. For grouping attributes into instances, we mention two basic approaches.

A. Instance parsing

This is a broad family of approaches, which consist in taking the results of some extraction method and trying to match the extracted attribute candidates against some extraction model (ontology). The extraction model defines classes; these classes typically have a number of constraints, which need to hold for each instance of the class.

The matching algorithms are based on different principles: Embley [7], [8] uses extraction ontologies containing regular expressions used to match the content and context of extracted attributes. The *Ex* system [16] takes a similar approach but incorporates probabilities into the extraction process, enabling both the use of training data and – where training data are not available – the use of manually specified patterns accompanied with precision and recall estimates. The resulting attribute candidates are scored which allows for parsing them into instances using dynamic programming algorithms. Over non-stochastic approaches this has the benefit of making possible trade-offs between local and global decisions during the parsing process. Techniques based on Bayesian networks [27] have also been applied successfully to both named entity and relation extraction.

B. Wrappers

In today's web, a large portion of documents within a certain website is generated dynamically based on a single template, leading to the same formatting. When extracting from a limited number of websites, and especially if the extraction process is scheduled to repeat, it is today a common practice to define manually or induce interactively sets of rules (wrappers) to extract the desired data just based on formatting. The wrapper rules can naturally extract whole instances consisting of multiple attribute values.

Wrapper-based approaches have the advantage of a single method which covers both the label and instance extraction problems. The disadvantage lies in that wrapper rules need to be created for each website separately and also they are fragile in terms of any changes to the website's formatting. Different approaches to wrapper induction are described e.g. in [13], [14], [15]. Wrapper reconstruction [21] is an approach aiming to detect formatting changes in the wrapped website and to re-create wrappers automatically. There also has been progress in the area of combining traditional IE machine-learning techniques with wrapper induction [25].

2.3 Information integration

When extracting data from texts, the task of information integration is to compare two extracted items and determine whether they are the same. Two items are usually defined to be "the same" if they correspond to the same real object. For example, IBM, I.B.M. and International Business Machines all refer to the same company and we may want to have it just once in the extracted item database. Information integration thus often involves a canonicalisation step where each extracted item is converted into its canonical form before being stored.

Information integration methods typically rely on surface similarities between the compared items. In all cases, different string matching techniques² are used to determine whether two different string values have the same meaning or not. A range of trainable string matching techniques is described e.g. in [1]. In MEDIEQ, we will use information integration techniques in at least the following cases:

- canonicalisation of extracted items prior to storing,
- determining whether a change between the currently extracted item and its previous version is important (changes meaning),
- during instance extraction, information integration is needed to find out whether an extracted attribute has multiple different values or whether there is the same value mentioned multiple times.

2.4 Information extraction tools

Today there are a number of IE tools which can be used to perform IE tasks of various complexities. Most available tools allow for named entity

² <http://sourceforge.net/projects/simmetrics/>

recognition, both based on training data and on manually specified patterns (e.g. regular expressions). Instance extraction seems to be supported to a somewhat lesser extent. Most available IE tools concentrate on extraction from free text and they often do not utilise web page structure and formatting clues. This does not hold for wrapper maintenance and induction systems such as Lixto³ or Kapow⁴ which in turn concentrate on wrappers only. For a reference survey of tools we may utilise in the IE toolkit, see Appendix B.

³ <http://www.lixto.com/>

⁴ <http://www.kapowtech.com/>

3. The information extraction methodology

All information extraction tasks in MedIEQ will be handled by the Information Extraction Toolkit (IET), which will consist of several document-processing components, some of which will provide a user interface.

3.1 Hierarchy of tools and components

A component tree of the IE toolkit is shown below.

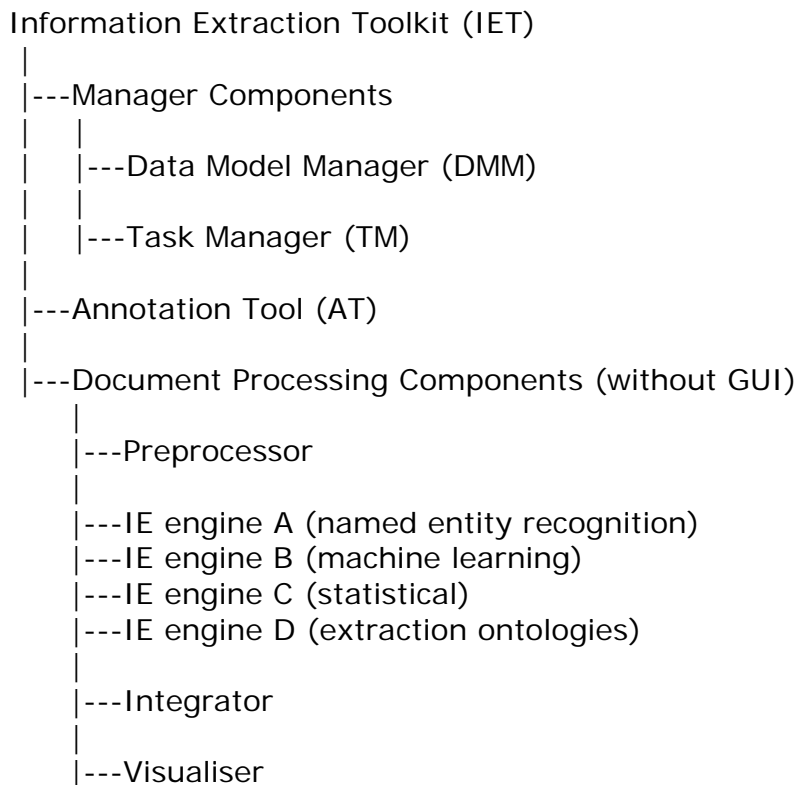


Fig. 1. Tree of components and tools constituting the IE Toolkit

From the user's perspective, there are two main components with user interfaces in the toolkit: the Data Model Manager (DMM) and the Task Manager (TM). DMM is designed primarily for administrators; it manages Extraction models and allows for adding extraction knowledge to them. Labelling experts will use TM to execute, monitor, evaluate and retrieve data from extraction tasks.

3.2 Tools and components quick reference

In the following we briefly describe the roles of each component in the toolkit.

Data Model Manager

DMM is the place where *extraction models* are stored. Extraction model is a set of extractable attributes and/or extractable classes. Extraction model is typically derived from a labelling scheme, supplied by the Label Management Toolkit (WP4). The labels are types of information to be extracted derived from machine-identifiable quality criteria. Each label will typically correspond to one extractable attribute, but for IE purposes, it may sometimes be useful to map one label to more attributes or vice versa. In addition, attributes that logically belong together are grouped into classes. For example, "person name" and "address" form a class "person" which has a cardinality constraint of having just one name.

In addition to containing the attributes and classes, an extraction model may link to *extraction knowledge* for each attribute or class. Extraction knowledge can be induced from training data or it can come from labelling experts (e.g. in the form of rules, constraints, value sets or patterns). DMM is responsible for the maintenance and editing of Extraction models and for providing them to IE engines. Its user interface will be primarily intended for use by administrators.

Task Manager

Task manager will be the central component allowing labelling experts to routinely use the IET. It will allow users to configure and run IE tasks with selected Extraction model(s) on selected sets of documents. It should monitor progress of running tasks, evaluate results and keep task history.

Using Task manager's UI, the users will be able to run documents through document processing components: the Preprocessor, selected IE engines and the Integrator. These components will perform extraction (using the task's Extraction model) by adding attribute annotations and/or extractable class instances to processed documents.

For each task, there will be options allowing the user to control the flow of documents through the document processing components. One option is to form a pipeline and run a single document sequentially through all IE engines, so that the latter engines may utilise annotations added by previous engines (the same concept is implemented in IBM's UIMA [9]). The opposite option would be to process each document in parallel (independently) by all engines, which would produce independent annotations from each engine. The Integrator will in both cases resolve possible disagreements in annotations produced by the different IE engines. Practical configurations may exist between the above extreme options, such as processing the documents by a named entity recogniser first and then running other IE engines independently.

Preprocessor

This component is responsible for common pre-processing of documents independent of specific IE engines. Pre-processing steps include:

1. conversion from the original document format (HTML and text in the first version; PDF, RTF and DOC subsequently),
2. performing any common changes to document source,

3. adding annotations to document.

The annotations added may be of two kinds. First, the Preprocessor should annotate document segments with generic information potentially useful to all IE engines. This information could contain results of running selected language processing tools such as a lemmatiser, part-of-speech tagger or a generic named entity recogniser.

Second, the Preprocessor will insert annotations describing any attributes and/or instances that have been extracted previously from the analysed document (if any). To accomplish this, the Preprocessor will interface with the MedIEQ's repository.

IE Engine

IE engine is responsible for adding to analysed document(s):

1. annotations corresponding to attribute definitions defined in Extraction model,
2. instances of classes composed of attributes found in document(s).

The first functionality is mandatory for each IE engine; the second is optional as extracting instances of classes having multiple attributes is not always necessary.

IE engines will provide functionality to label or extract instances from a single document, or from a collection of documents. Extracting from a collection is important for IE engines that can use extra-document information to improve extraction performance. For example, a document collection may consist of documents from a single website, documents of a certain class from a single website or documents satisfying a certain URL pattern from a single website. An IE engine capable of inducing wrapper patterns could then take advantage of common formatting structure exhibited by the whole document set.

See Appendix B for description of the MEDIEQ initial set of labelling criteria (details on these criteria are given in section 3 of Deliverable D4.1), and for the list of methods we intend to use for each of the labels.

As the initial set of labelling criteria contains diverse labels, we will need to employ different IE engines, each of which will implement a different extraction method.

For some of the labels a particular method has already been selected, while for other labels some experimentation will be needed to select the best method. Typically, we will assign each label to a single extraction engine, but in some cases, we may want to extract the same label using multiple engines and then to integrate results to achieve higher precision and recall. For this purpose, we have proposed the Integrator component which will be implemented if proven useful.

Based on the initial label set and also on the research interests of the involved partners, we chose to implement the following IE engines.

A. Named Entity Recognition (NER)

This component should identify all relevant named entities it can, even when they may not directly be subject to extraction. Documents annotated by NER will feed the other IE engines which may utilise the annotated named entities.

NER will combine a set of language pre-processing modules along with different techniques for named entity recognition in order to improve its performance. More specifically, NER will involve language pre-processing modules for tokenization, sentence splitting, part-of-speech tagging, stemming, and gazetteer lookup.

Concerning the named entity recognition techniques to be used, these will be mainly machine learning based (see below). For the purposes of MedIEQ, we will examine a combination of token-based approaches (e.g. the Trigrams 'n' Tags (TNT) tagger, the Brill tagger, the CRF++ toolkit), and possibly document-level approaches. For combination we will examine voting or stacking approaches. We will also use hand-crafted rules in order to extract standard entity forms such as e-mails etc.

NER component will also have the ability to extract the recognized named entities in structured XML files, which may be used by other components and toolkits.

B. Machine Learning (ML)

Machine Learning may be used in a variety of ways to extract useful pieces of information from a given resource. Active learning integrates IE algorithms into annotation tools, supporting on the one hand the work of human annotator and producing generalised extraction rules that can be applied then to new documents. An alternative way to use ML for information extraction is to reduce the extraction problem into a classification one. Meta-learning techniques can also be used that combine the results of different extraction techniques.

Taking into account the specific requirements of the MedIEQ labelling criteria, different ML techniques will be examined. For this purpose, we will exploit NCSR's existing infrastructure (Ellogon language engineering platform, its interfaces with the Weka machine learning library and other ML algorithms, as well as Ellogon tools for corpus annotation, use of ontologies, dictionaries, gazetteers, etc.).

C. Statistical Text Processing (STA)

Statistical processing of web resources' contents can be employed to extract various properties of texts, which can be difficult to detect with other approaches. Automatic keyword detection is such an example where an analysis on the word use frequencies in a document can be used to extract relevant keywords. Further, statistical analysis allows documents to be analysed in an unsupervised manner to group similar documents (websites) into clusters.

The primary advantage of using statistical methods in information extraction is the relaxed requirement for human work to analyse large amounts of text materials. The use of unsupervised methods also has the virtue of being mostly language independent – the same methods can be applied to all languages.

D. Extraction Ontologies (EXO)

In case of this IE engine, the extraction process consists of two phases:

- identification of named entities,
- search for groups of related named entities which constitute instances.

The first phase can benefit from knowing named entities which have already been identified by other IE engines, such as NER, ML or STA. The Ex system operates based on an *extraction ontology*, which is essentially a domain ontology augmented with extraction knowledge. An extraction ontology typically contains a single or a few *classes* consisting of *attributes*, various types of constraints and axioms. A domain expert with basic understanding of IE concepts is needed to develop such extraction ontology for a new domain.

A prototype, code-named *Ex*, is being developed and utilises three types of extraction knowledge:

- manually specified by labelling experts (in extraction ontology),
- induced from training data (induced pattern rules referred from extraction ontology),
- formatting patterns found in analysed document sets (during runtime).

For named entity identification, Ex combines known extraction patterns to create scored candidates for each attribute found in extraction ontology. We plan to use named entities identified by other IE engines as if they were additional extraction patterns identifying the content of the corresponding attribute.

During instance extraction, our goal is to find the combination of isolated named entities and instances which best explain the attribute candidates identified in the examined document. Each attribute candidate can be explained either as an extraction error, as a separate named entity, or as an attribute belonging to a wider instance. We utilise a bottom-up chart parsing approach to create scored instance candidates composed of two or more attribute candidates. During this process, we try to identify regular formatting patterns used in HTML pages and use them to boost scores of instance candidates which satisfy these formatting patterns, and to create additional attribute candidates. The parsing process takes into account constraints and axioms encoded in the extraction ontology. Finally, we search for the best path through individual attribute candidates and instance candidates to retrieve the resulting sequence of extracted items – these can be instances interleaved with separate named entities. More details about *Ex* can be found in [16], [17] and [19].

Integrator

Integrator should process all named entities and instances identified by the different IE engines in a document or a collection of documents and integrate them into a single set of extracted named entities and instances. This component will implement interfaces similar to IE engines; since it is itself an IE engine working on the basis of other engines' results (see [17] for the proposed interfaces implemented by the Integrator component).

The Integrator is useful only if the label sets extracted by the utilised IE engines overlap. Such overlaps may be desired to achieve higher precision and recall. However, if the labels extracted by different engines do not overlap, the Integrator can be left out. We will determine experimentally whether the Integrator component will be needed.

As the first approach, we will connect the Preprocessor and the four IE engines (A, B, C and D, respectively) in a pipeline so that subsequent engines can use the information identified by their predecessors. In this setup, the Integrator's functionality will be embedded in the last IE Engine in the pipeline – the Extraction Ontology engine.

Visualiser

The Visualiser is a component responsible for rendering extracted items in processed web pages. Different rendering ways are currently being examined by UEP and NCSR.

Annotation Tool

Training data for trainable IE engines is being obtained using a visual annotation tool based on the Ellogon language engineering platform from NCSR. UEP and NCSR are currently examining a way to provide an initial version of the training data automatically to save the work of human annotators.

4. The Information Extraction Architecture

This section describes the APIs exposed by the IET components and shows how these components may communicate with each other and with the rest of the MedIEQ system. As in the case of user interfaces, the IET is designed to provide its functionality using the two manager components – the Data Model Manager and Task Manager. For a high-level view of the components and basic document flow, see Fig.1.

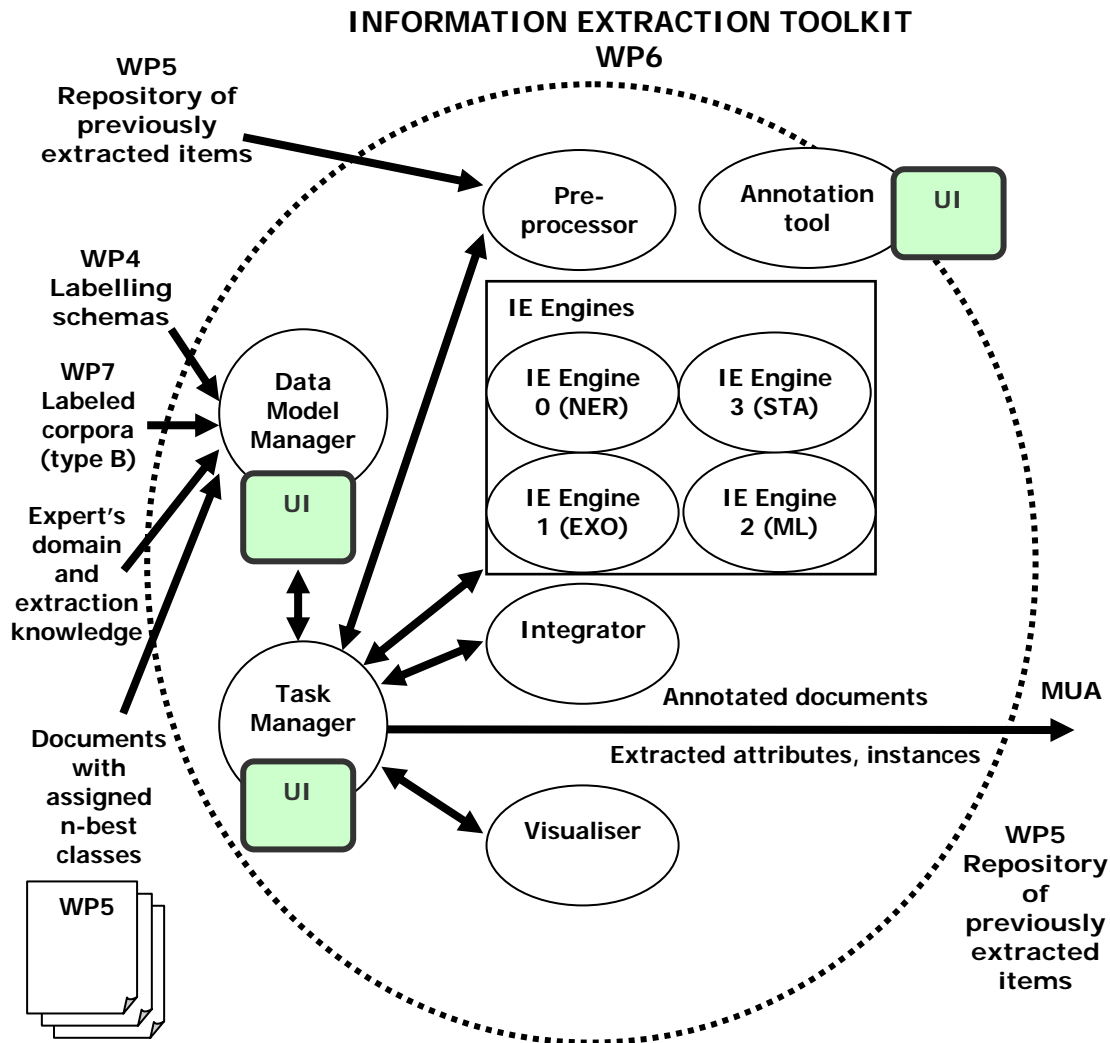


Fig.1. IE Toolkit Architecture

4.1 Manager Components APIs

Both the DMM and TM implement the Configurable interface, which allows for common configuration procedure. For an overview of proposed DMM and TM methods, see [17].

The **DMM** offers methods for loading and saving Extraction models and for adding extraction knowledge to them. Editing of the Extraction models will be done using the DMM GUI.

The *TM's* API allows for creating, manipulating and running IE Tasks. A single IE Task is modelled by a separate class (***Task***) which is part of the public API. An IE Task encapsulates a set of documents to be processed, and an Extraction model according to which extraction will be done. When a client configures and runs an IE Task, it can register a listener (***TaskListener***) to receive events (***TaskEvent***) announcing the task's progress, including any errors. The TM also has a GUI, which allows users to perform the same operations as offered by the API.

For web sites which have already been labelled, the TM will work in close cooperation with the Web Content Collection Toolkit. The TM will offer an option to query the WCC only for documents which have changed significantly since the last run. For all modified documents, the previously extracted items will be retrieved and will be accessible for viewing and comparison within the IE Task.

4.2 Data Container Classes

In IET, there are several important classes which are communicated between the toolkit's components. These classes also determine what information can be communicated with other MedIEQ toolkits, especially the WCC, LAM, MRM, MUA and the Database. For the content of these data container classes and relations among them, see [17].

The central class is ***Document***, which encapsulates the document source, metadata and an n-best array of classes produced previously by the WCC's document classifier. The Document object flows through the IET's document processing components which may perform the following operations:

- The Preprocessor may change the document source (its locally cached content).
- IE Engines may add annotations of attributes by identifying an ***AttributeValue*** in document source and by adding an ***Annotation*** object which links to that AttributeValue. There may be more Annotations linked to a single AttributeValue since an extracted attribute can be mentioned repeatedly.
- IE Engines may add instances (valid groups of AttributeValues which typically do not overlap).
- The Integrator may modify or delete annotations and corresponding extracted attributes and instances.
- The Visualiser adds mark-up to document source in order to show the extracted attributes and instances.

The AttributeValue class links to its definition, the ***AttributeDef*** class. Objects of this class are contained in the ***ExtractionModel***. Attribute definitions may either be independent or they can be part of one or more class definitions (***ClassDef***). Both attrib and class definitions typically contain specific extraction knowledge used by IE Engines.

4.3 Data Processing Components

Data processing components (*Preprocessor*, *IE Engines*, *Integrator* and *Visualiser*) all share a certain part of API. We try to identify this common functionality and abstract it into a series of interfaces, shown in [17]. The interfaces are:

- Configurable,
- DocumentModifier,
- LabelExtractor,
- InstanceExtractor,
- Trainable,
- DocumentAnnotator.

The **Configurable** interface is a common interface allowing for common configuration of all components in the toolkit.

The **DocumentModifier** is to be implemented by components which need to change the document's source. So far this is just the Preprocessor.

The interfaces **LabelExtractor** and **InstanceExtractor** are to be implemented by IE Engines; the latter only by those which extract instances. Both interfaces do not allow the document source to be changed, they only may add, change or remove annotations and instances. Both interfaces will also be implemented by the Integrator, as it will need to consolidate the annotations and instances added previously by independent IE engines.

The **Trainable** interface can be implemented by all document processors which can learn their specific extraction models from annotated training documents. These may include the IE Engines and the Integrator.

The **DocumentAnnotator** will be implemented by the Visualiser component which will visually annotate extracted attributes and instances in the original document source so that they can be viewed in a web browser. Technical report [17] shows which IET components implement which interfaces.

5. Users, user interfaces and use cases

5.1 Users & UIs

The two user roles in MedIEQ are the labelling expert (a quality specialist from AQUMED or WMA) and the system administrator. We will use LE to refer to the labelling expert and SA to refer to the system administrator. The actions available in the toolkit's user interfaces will be intended for both or just one of these roles.

5.2 Use cases

Actors & interactions

An actor is the user of the MEDIEQ system. Actors may be both humans and computer programs. These software (SW) actors include other components from this or other toolkits or external systems. Below, we enumerate all actors relevant to the IE toolkit:

Human actors

1. Labelling expert (an expert from AQUMED or WMA)
2. System administrator

SW actors

1. LAM (Label Management Toolkit, the WP4 toolkit)
2. WCC (Web Content Collection, the WP5 toolkit)
3. IET (Information Extraction Toolkit, the WP6 toolkit)
4. MRM (Multilingual Resources Management Toolkit, the WP7 toolkit)
5. MUA (Monitor-Update-Alert, the WP8 toolkit)
6. MedIEQ Database
7. Scheduled jobs

Notice that we distinguish between two interaction levels:

- High Interactions Level [-H-]: where all the above listed actors can directly interact with each other. This means that toolkits interact with other toolkits, the database and the scheduled jobs. Every toolkit has an API. The specified API actions are linked to interfaces.
- Low Interactions Level [-L-]: within this level we group the interactions between components inside a toolkit. Tools and components inside a toolkit interact with each other according to the toolkit internal architecture.

Use case descriptions

In the following text we describe the major use cases of IET. As only the two manager components have graphical interfaces accessible by users, all use cases only involve the DMM and TM. A complete UML use case diagram of the IET is shown in [17].

DMM will allow administrators to create, manage and edit Extraction models, define attributes and classes in the Extraction model, and attach training data to attribute and class definitions.

TM will host a user interface allowing users to run IE tasks with selected Extraction model(s) on selected document sets. It will be able to monitor progress of the executing task(s) and keep track of task history. It should provide achieved precision, recall, elapsed time and other task information. IE task will be configurable to include chosen IE engines and will manage parameters for the Preprocessor, the IE engines and the Integrator.

Other components – Preprocessor, IE engines and Integrator – will offer system administrator UI & API only. These components are not required to offer convenient user interfaces. In the following, we use SA for system administrator and LE for labelling expert.

Use case 1: Derive a new Extraction model (or edit existing)

Actors

[-H-] System administrator, IET, LAM

[-L-] DMM

Interactions

[-H-] SA imports a labelling schema (a set of labels) from LAM, creating a new Extraction model (a set of attribute definitions which originally directly correspond to labels).

[-H-] SA makes any changes to the model necessary to make it suitable for information extraction. This may involve adding data types, attribute splitting or merging, grouping attributes into logical classes (for instance extraction), defining attribute cardinalities.

Use case 2: Add extraction knowledge to an Extraction model

Actors

[-H-] System administrator, IET

[-L-] DMM

Interactions

[-H-] SA opens an extraction model.

[-H-] SA links elements of the extraction model (attributes or classes) to any of the two extraction knowledge types: 1) human expert extraction knowledge, 2) knowledge induced from training data.

[-H-] SA saves the extraction model.

Use case 2.1: Add human expert extraction knowledge

Actors

[-H-] System administrator, IET, WCC, Database, MRM

[-L-] DMM

Interactions

[-H-] SA chooses a single Extraction model attribute corresponding to a certain label.

[-H-] SA queries WCC for documents containing the chosen label and views their contents.

[-H-] SA queries Database for previously extracted examples of the label.

[-H-] SA browses the MRM interface to find any existing resources (e.g. value lists) that could be used to extract the chosen attribute.

[-H-] SA defines and links new extraction knowledge to the chosen attribute. SA may e.g. define an extraction pattern using regular expressions on words or letters, link to an external value list, impose minimum and maximum constraints for numeric types etc.

[-H-] SA may define axioms which constrain possible values of a certain attribute, or define relationships between related attributes.

Use case 2.2: Add extraction knowledge from training data

Actors

[-H-] System administrator, IET, WCC, MRM, Database

[-L-] DMM, IE Engine, TMG

Interactions

[-H-] SA chooses a single attribute corresponding to a certain label definition.

[-L-] A. DMM queries WCC for training documents containing annotated values of this label.

[-L-] B. DMM queries WCC for training documents which belong to a relevant document class (for the chosen label) but do not contain annotated values.

[-L-] C. DMM queries Database for previously extracted values of the chosen label (we assume these are linked to the source document).

[-L-] D. DMM enumerates extraction knowledge already present in Extraction model as entered previously by extraction experts.

[-L-] DMM invokes a training component with the training data collected in the above steps A.–D. The training component can either be directly one of the IE engines, or we might reuse the generic TMG component.

[-H-] SA views training results (specific to each IE engine).

Use case 3: Define, run and evaluate an extraction task

Actors

[-H-] System administrator, IET

[-L-] Task Manager (TM)

Interactions

[-H-] SA sets options for the extraction task and whether previously extracted items from the same document should be retrieved for comparison (if available).

[-H-] SA saves the Extraction model and the document set as an extraction task.

[-H-] SA runs the extraction task and monitors its progress. Multiple tasks may be run in parallel.

[-H-] SA browses history of extraction tasks maintained by TM.

[-H-] SA can view the achieved precision and recall for extraction tasks which were run on documents which already contained annotated labels.

Use case 4: View extracted data and make it accessible to others

Actors

[-H-] Labelling expert, IET, Database

[-L-] Task Manager (TM)

Interactions

[-H-] LE browses through and views all instances (groups of labels) or all individual labels extracted by a chosen extraction task.

[-H-] LE browses through and views documents annotated by a chosen extraction task (also handled by the Label Management Toolkit).

[-H-] LE compares extracted items with those extracted previously for the same document (where "the same" means retrieved from the same address or otherwise as defined by WP5).

[-L-] TM accepts queries from Database which will further handle the extracted instances or labels.

6. Concluding remarks

In this document we described the architecture of the Information Extraction toolkit and outlined a methodology for using it by different types of users in combination with other MedIEQ toolkits.

As we are currently implementing most of the IE Toolkit's components, the presented architecture is still subject to change as we identify new challenges during both the development and testing stages.

The architecture should however be robust with respect to changes in the labelling schema, including the addition of more labelling criteria.

The architecture enables new IE engines to be plugged in and combined with existing IE engines, which is necessary in order to support new labelling criteria as well as the addition of new languages.

Bibliography

- [1] M. Bilenko and R. Mooney. Employing trainable string similarity metrics for information integration. In: IJCAI Workshop on Information Integration on the Web 2003.
- [2] V. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records. In: SIGMOD 2001.
- [3] M.E. Califf and R.J. Mooney. Bottom-up relational learning of pattern matching rules for information extraction. In: AAAI 1998.
- [4] F. Ciravegna. Adaptive Information Extraction from Text by Rule Induction and Generalisation. In: IJCAI 2001.
- [5] F. Ciravegna. LP2, an Adaptive Algorithm for Information Extraction from Web-related Texts. In: IJCAI-2001 Workshop on Adaptive Text Extraction and Mining held in conjunction with the 17th International Joint Conference on Artificial Intelligence, 2001.
- [6] M. Collins and B. Roark: Incremental Parsing with the Perceptron Algorithm. In: ACL 2004.
- [7] Y. Ding, D.W. Embley and S.W. Liddle: Automatic Creation and Simplified Querying of Semantic Web Content: An Approach Based on Information-Extraction Ontologies. In: ASWC 2006.
- [8] D.W. Embley, C. Tao, and S.W. Liddle. Automatically extracting ontologically specified data from HTML tables with unknown structure. In: ER 2002.
- [9] D. Ferrucci and A. Lally. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering* 10, No. 3-4, 2004.
- [10] A. Finn and N. Kushmerick. Active learning selection strategies for information extraction. In: ECML-2003 Workshop on Adaptive Text Extraction and Mining, 2003.
- [11] D. Freitag and A. McCallum. Information extraction with HMMs and shrinkage. In: AAAI Workshop on Machine Learning for IE 1999.
- [12] D. Freitag and A. McCallum. Information Extraction with HMM Structures Learned by Stochastic Optimization. In: AAAI/IAAI, 2000.
- [13] C.A. Knoblock, S. Minton, J.L. Ambite, N. Ashish, P.J. Modi, I. Muslea, A.G. Philpot, and S. Tejada. Modeling Web Sources for Information Integration. In: AAAI WI 1998.
- [14] N. Kushmerick, B. Thomas: Core Technologies For Information Agents. In: LNCS, 2003.
- [15] N. Kushmerick, D. S. Weld, and R. Doorenbos. Wrapper Induction for Information Extraction. In: Intl. Joint Conference on Artificial Intelligence 1997.
- [16] M. Labsky and V. Svatek: On the Design and Exploitation of Presentation Ontologies for Information Extraction. In: Workshop Mastering the Gap at ESWC 2006.

- [17] M. Labsky: Information Extraction Toolkit UML models, technical report, <http://eso.vse.cz/~labsky/doc/iet>, 2006.
- [18] M. Labsky, V. Svatek, O. Svab, P. Praks, M. Kratky, and V. Snasel. Information Extraction from HTML Product Catalogues: from Sorce Code and Images to RDF. WI 2005.
- [19] M. Labsky and V. Svatek: Information Extraction with Presentation Ontologies, whitepaper, <http://eso.vse.cz/~labsky/doc/ex.pdf>, 2006.
- [20] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: ICML 2001.
- [21] K. Lerman, S.N. Minton, C.A. Knoblock: Wrapper Maintenance: A Machine Learning Approach. In: Journal of Artificial Intelligence Research 18, 2003.
- [22] C.D. Manning and H. Schuetze. Foundations of Statistical Natural Language Processing. The MIT Press, 2000.
- [23] A. McCallum, D. Freitag and F. Pereira: Maximum Entropy Markov Models for Information Extraction and Segmentation. In: ICML 2000.
- [24] T.M. Mitchell. Machine Learning. McGraw-Hill, 1997.
- [25] M.T. Paziienza, A. Stellato, and M. Vindigni. Combining ontological knowledge and wrapper induction techniques into an e-retail system. In: ECML/PKDD workshop ATEM 2003.
- [26] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In: IEEE 1989.
- [27] D. Roth and W. Yih: Probabilistic Reasoning for Entity & Relation Recognition, COLING 2002.
- [28] I. Schroeder. A Case Study in Part-of-Speech Tagging Using the ICOPOST Toolkit. Technical Report, Department of Computer Science, University of Hamburg, 2002.
- [29] F. Sha and F. Pereira: Shallow parsing with conditional random fields. In: HLT-NAACL 2003.
- [30] A. De Sitter and W. Daelemans: Information Extraction via Double Classification. In: ATEM 2003.
- [31] K. Takeuchi and N. Collier: Use of Support Vector Machines in Extended Named Entity Recognition. In: CoNLL 2002.
- [32] I.H. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann 1999, 1-55860-552-5.

APPENDIX A. Terminology

Information extraction terminology

Attribute (definition)	Single well-defined type of information to be extracted; e.g. a person name or degree, last modified date, free-form symptom description. Typically corresponds to a single <i>Label</i> , but for IE purposes, <i>Label</i> can be split into more attributes or vice versa.
Attribute value	Specific value of an Attribute, found in a document or in a database of extracted items. May be canonicalised before being stored in database.
Annotation	Consecutive segment of a document, annotated by some IET component (during an extraction task) or by a human (in case of training documents). Attribute value is one example of an annotated object; other example can be a noun phrase.
Class (definition)	A logical group of Attributes, e.g. a person that consists of a person name and address.
Instance (of a Class)	Specific example of a Class, found in a document or in a database of extracted items.
Extraction model	A set of Class and Attribute definitions, where both may be associated with extraction knowledge.

Related terminology

Criterion	A feature of a medical website whose value is significant for telling the quality of that website. Some criteria are <i>machine-identifiable</i> , such as the presence of the last modified date or identification of a responsible medical professional. Such machine-identifiable criteria can be translated into one or more labels (e.g. medical professional's name, degree and address) which we attempt to find automatically.
Label	Extractable item derived from a criterion.
RDF label	Label encoded in RDF format.

APPENDIX B. Initial set of labelling criteria

The abbreviations used include:

ML machine learning (statistical or rule-based)

NER named entity recognition using patterns, dictionaries, gazetteers

HEU ad-hoc heuristics

By document, we mean both a web page and any other document regardless of its format.

1. RESOURCE TITLE

Problem type: extraction

Applies to: document, web site

Extraction methods (document):

1. HEU (<title>...</title> field for HTML pages, other heuristics for other document types).
2. Keyword extraction techniques when no title available.

Extraction methods (web site):

1. HEU (<title>...</title> field of the home page)

2. RESOURCE URI

Problem type: extraction

Applies to: document

Extraction methods (document):

1. The Spider component from the WCC will extract URI.

3 & 4. CONTACT DETAILS (FOR AUTHOR / RESPONSIBLE)

Contact information includes: person name, organization, department, street address, city, county, country, phone number/extension and e-mail.

Problem type: extraction

Applies to: document, web site

Extraction methods (document – AQuMed approach):

We search for the author of the resource.

Step 1: Extraction of attribute candidates

Person name, organization, department, street address, city, county, country:

1. NER + ML

Phone number/extension and email:

1. NER, maybe ML

For English and Czech we have collected large dictionaries of first names, last names, state/county names, city names, street names and zip codes. This data is being used in our current experiments with NER.

Step 2: Instance extraction

This step selects high-scoring combinations of attribute candidates and groups them into instances of an "Author" class. Within *Ex*, a bottom-up parser is used which searches for the best way of combining attribute candidates into instance candidates, subject to class constraints. After generating scored instance candidates, the best path through the document consisting of non-overlapping valid instance candidates is found.

Step 3: Decide which of the extracted Author instances really are the authors of the document.

1. When attribute candidates are extracted, their score is boosted by the presence on contextual clues correlated with the presence of author. The scores of the attribute candidates are reflected in the instance score; therefore one approach is to select the top-scoring Author instances.
2. Extract generic Person instances and then look at the contextual clues using an independent ML method.

In the future we may need to identify multiple overlapping roles like authors of content, contact persons, responsables, medical specialists, webmasters etc.

Extraction methods (web site – WMA approach):

The goal is to find the person responsible for the content of the entire web site. We repeat the steps 1-3 mentioned above for each candidate contact page in the web site. The class to be extracted is a modified version of "Author"; we will refer to it as "Responsible". It will primarily differ from Author in knowledge representing characteristic textual contexts. If multiple Responsibles are found, we need to find out which should be returned as the web site's responsables.

1. Based on the Responsible instance score and the score of the source document being classified as "home page" or contact page", compute a combined score and then return the best instances.

5. LAST UPDATE

Problem type: extraction

Applies to: document, web site

Extraction methods (document):

1. Heuristics and NER for extraction of candidate dates found in the document, ML to decide whether the date represents last update or not. Feature vector will use a number of tokens before and after the extracted date.

2. Server last modified date. This is easy but we will often get the current time when processing dynamically generated pages.

Extraction methods (web site):

1. Use the last update of home page.
2. Use the last update all pages' last updates within the website.

6a. KEYWORDS (MeSH)

Problem type: extraction

Applies to: document, web site

Extraction methods (document):

1. Words whose lemma does not occur in the MeSH thesaurus are filtered out. Most important keywords will be selected using the TF/IDF method.
2. HTML meta tag "keywords", which however may not be present and sometimes may be misused.

Extraction methods (web site):

1. Lemmatised MeSH terms with the greatest TF in the entire web site will be selected as keywords.
2. HTML meta tag "keywords" found in the home page.

For MESH keyword extraction we are currently using a subset of terms extracted from the English and Czech versions of the MESH database. For English we do not use a lemmatiser, for Czech we do. The final subset of MeSH keywords needs to be clarified.

6b. TOPIC (MeSH)

Problem type: classification

Applies to: document, web site

Classification methods (document):

1. A Web Page will be classified to one or more categories according to its keywords based on the MeSH categorisation.
2. A Machine Learning algorithm will classify each Web Page. As features (in the feature vector) we can use either the keywords of the page extracted with the method described above or another feature selection method (an annotated corpus is necessary).

Classification methods (web site):

1. Use the union of the topics found in all web pages.
2. Use at most N of the prevailing topics.

The final set of MeSH topics needs to be clarified; a starting point could be to use all nodes of the MeSH hierarchy of a certain depth.

7. RESOURCE LANGUAGE

Problem type: classification

Applies to: document, web site

Classification methods (document):

1. ML classification: character histogram models for known languages.
2. ML classification: histograms of most common words for known languages.
3. Document source encoding or language identifiers such as HTML attribute "lang" (typically unavailable).

Classification methods (Web Site):

1. Return the union of languages found in all pages.
2. Return the prevailing language.

8. TARGET AUDIENCE

Problem type: classification

Applies to: document, web site

Extraction methods (document):

1. ML trained classifier for categories adult, children and professionals.
Feature set could include word uni/n-gram histogram.

Classification methods (web site):

1. Union of target audiences of all pages.
2. Prevailing audience.

9. VIRTUAL CONSULTATION

Problem type: classification

Applies to: document, web site

If one web page offers a VC service then the web site offers VC service.

Classification methods (web site):

1. Machine Learning (classification)
2. Heuristics (e.g. patterns defined on HTML form elements and surrounding keywords)
3. Combination of the above two

Extraction methods (web site):

1. Union of identified virtual consultation pages.

10. ADVERTISEMENTS PRESENT

Problem type: classification

Applies to: document, web site

If one web page contains an advertisement then the web site contains advertisements.

Classification methods (document):

1. ML classification algorithms with a rich set of features. Features may represent all tokens and HTML tags within document and characteristics of image content (e.g. histograms or recognised text) and sizes [18].
2. HEU (e.g. regular expressions)
3. Combination of the above two.
4. Database with known advertisement web sites.

Extraction methods (web site):

1. Union of advertisements found in documents.

11. OTHER SEAL

Problem type: extraction

Applies to: document, web site

If one web page contains a seal then the web site is sealed.

Extraction methods (document):

1. HEU (e.g. regular expressions) for real seals
2. Wrappers for virtual seals (AQUAMED, OMNI, WHO)

Extraction methods (web site):

1. Union of seals extracted from documents

APPENDIX C. Extraction tool survey

LingPipe

URL: <http://www.alias-i.com/LingPipe>

web demo: <http://alias-i.com:8080/LingPipe-demos/>

LingPipe is a suite of Java libraries for the linguistic analysis of human language. The API is well documented⁵ and the source code is freely available, as well as a sandbox repository. In general the tool it's easy to use and quite didactic⁶, either through sample code or command line demo executables.

In particular the NER component is based on Hidden Markov Models. The model that comes bundled with the LingPipe distribution has been created using the MUC6⁷ English corpus. It should be stressed that any serious use of the NER component should train its own model. The MUC6 model is based on a very small amount of data compared to e.g. the ACE dataset. Both training and evaluation components in LingPipe are easy to use. The co-reference module is still under development.

LingPipe's information extraction and data mining tools:

- track mentions of entities (e.g. people or proteins);
- link entity mentions to database entries;
- uncover relations between entities and actions;
- classify text passages by language, character encoding, genre, topic, or sentiment;
- correct spelling with respect to a text collection;
- cluster documents by implicit topic and discovers significant trends over time; and
- provide part-of-speech tagging and phrase chunking.

LingPipe's architecture features:

- Java API with source code and unit tests;
- multi-lingual, multi-domain, multi-genre models;
- training with new data for new tasks;
- n-best output with statistical confidence estimates;
- online training (learn-a-little, tag-a-little);
- thread-safe models and decoders for concurrent-read single-write (CRSW) synchronisation; and
- character encoding-sensitive I/O.

GATE (Annie)

URL: <http://gate.ac.uk/annie>

Web demo: <http://gate.ac.uk/annie/>

⁵ <http://www.alias-i.com/LingPipe/docs/api/index.html>

⁶ See tutorials at <http://www.alias-i.com/LingPipe/demos/tutorial/read-me.html>

⁷ MUC6 website available at <http://cs.nyu.edu/cs/faculty/grishman/muc6.html>

ANNIE is designed to be a Portable IE system. In other words ANNIE is intended to be usable in many different applications, on many different kinds of text and for many different purposes.

ANNIE is implemented in Java, and it is embedded into a larger infrastructure: GATE (General Architecture for Text Engineering). GATE has extensive documentation⁸ and some useful examples. It should be noted that in general the learning curve for GATE/ANNIE seems to be harder than for some other systems.

ANNIE NER module is based on rules (using the JAPE language) and gazetteers. The usual process pipeline is composed of the following modules: tokeniser, sentence splitter, POS tagger, NE tagger and orthographic/pronominal co-reference. In particular the co-reference modules add extra slowness to the NER process.

BiOs tagger

URL: <http://www.lsi.upc.es/~surdeanu/bios.html>

BiOs is a suite of syntactic/semantic analysers that include the most common tools needed for the shallow analysis of English text. The following tools are currently included:

- Smart tokeniser that recognises abbreviations, SGML tags etc.
- Part-of-speech (POS) tagger. The POS tagger is implemented as a wrapper around the TNT Tagger⁹ by Thorsten Brants.
- Syntactic chunking using the labels promoted by the CoNLL chunking evaluations¹⁰.
- Named-Entity Recognition and Classification (NERC) for the CoNLL¹¹ entity types plus an additional 11 numerical entity types.

It can be configured for very high accuracy but slower execution (using Yamcha¹²) or for high speed and slightly lower accuracy (using an asymmetric Perceptron). It has built-in models for both case-sensitive and case-insensitive text.

The bundled NE model is based on the English CoNLL dataset. The included NERC recognises the following entity types:

- PER - person names (trained on CoNLL).
- LOC - location names (trained on CoNLL).
- ORG - organization names (trained on CoNLL).
- MISC - miscellaneous other names (trained on CoNLL).
- MONEY - monetary expressions (recognised with regular expressions).
- DATE - temporal expressions (regular expressions).
- LANGUAGE - names of languages (regular expressions).
- PERCENT - percentage expressions (regular expressions).

⁸ <http://gate.ac.uk/sale/tao/index.html>

⁹ <http://www.coli.uni-saarland.de/%7EThorsten/tnt/>

¹⁰ <http://www.cnts.ua.ac.be/conll2000/chunking/>

¹¹ <http://www.cnts.ua.ac.be/conll2002/ner/>

¹² <http://chasen.org/%7Etake/software/yamcha/>

- DISTANCE_QUANTITY - linear distance measures (regular expressions).
- SPEED_QUANTITY - speed measures (regular expressions).
- TEMPERATURE_QUANTITY - temperature measures (regular expressions).
- SIZE_QUANTITY - size measures (regular expressions).
- WEIGHT_QUANTITY - weight measures (regular expressions).
- ANGLE_QUANTITY - geometric angle measures (regular expressions).
- NUM - generic numbers not included in any other expressions.

Also, it can be re-trained with different corpus and label set. It has a clean and easy to use Java API, but not as much documentation as would be desirable.

JET

URL: <http://cs.nyu.edu/grishman/jet/jetDownload.html>

Documentation: <http://cs.nyu.edu/grishman/jet/doc/Jet.html>

JET, the Java Extraction Tool, provides a variety of components for language analysis. These components can be arranged in pipelines for different applications, and can be used either for interactive analysis of individual sentences, or 'batch' analysis of complete documents. JET is a work in progress, and is being regularly expanded and updated.

In general this can be considered the most updated tool from a research point of view. It's constantly developing, but most of this new developments are still unreleased to the public.

The NE component is trained using the ACE dataset which is considerably larger than the tagged corpus of CONLL and of course MUC6.

Further information

A more extensive description of NER systems and NLP frameworks with NE capabilities, visit

<http://alias-i.com/LingPipe/web/competition.html>.