Part II

Analysis

7 Aims and Requirements

After having covered extensively the current state of the art in the field of information extraction, we are now ready to formulate the specific aims of our own work. We will return to each of the aspects discussed in the previous chapter, with the goal of identifying aims and requirements for our own work. Our general guiding principle is to identify and preserve the best and most promising techniques from current approaches while, at the same time, exploring issues and investigating problems that so far have been neglected.

7.1 Aims of Our Approach

7.1.1 Primary Task to Handle

In Section 3.1 we have seen that a comprehensive algorithm for populating a database with information extracted from text documents will generally comprise various steps, but that one only of these steps—the *extraction of explicit information* (fragment extraction)—is required in all cases. And in Section 6.1 we have seen that most current IE systems handle only this one step, while those few that also handle *relationship recognition* often do so in a very limited way (resolving only relationships within a single sentence) or else use rule-based recognition mechanisms which appear to be somewhat ad-hoc and not necessarily suitable for other corpora. The *SNoW-IE* variant presented in [Rot02] (cf. Sec. 4.5) handles relationship recognition in a more principled fashion, but it, in turn, does not really handle the fragment extraction task, requiring the extracted text fragments to be already given.

There appears to a be clear tendency to concentrate efforts on a single step, and this might well be justified to avoid the loss of focus, considering that addressing all the steps outlined in Section 3.1 would be far too ambitious for a single work. Since the extraction of explicit information (fragment extraction) is certainly the core task for any information extraction system, we too will focus our work on this step, but with the understanding that is only one (though a very important one) step within the context of a more comprehensive solution that remains to be created.

7.1.2 Types of Texts to Handle

In Section 6.2 we have introduced the distinction between *free texts*, *semi-structured texts*, and *structured texts*. Structured texts are generally generated by computers, while both free and semi-structured texts are written by humans. This implies that it makes sense to use different algorithms for structured texts than for the other kinds of texts; the success of *wrapper induction* (Sec. 5.3) approaches on structured texts (but not on other texts) confirms this.

The area of structured text processing is already well researched and there is little point in writing yet another wrapper induction–like approach. Hence we will focus on human-written free texts and semi-structured texts instead of computer-generated structured texts.

However, our system should be suitable for *both* free and semi-structured texts, since both these kinds of human-written texts are important. There is an increasing amount of quickly written semi-structured texts where correct grammar and style are less of an issue, due to the success of text-based communication forms such as e-mail, newsgroups, Web forums and the like; but more formal free texts written for newspapers, press associations, governmental agencies etc. continue to remain important sources of information.

Our system should by a general-purpose information extraction system, it should not be tailored for a specific text type or domain (e.g. by containing domain-specific heuristics).

7.1.3 Features to Consider

In the last chapter (Sec. 6.3) we have seen that current IE systems use various kinds of features, typically based on tokens and word shapes, linguistic preprocessing, and semantic resources such as gazetteers. Since all of these features appear to be useful at least in some cases (otherwise they would not be used), our system should also be able to use them; but we will evaluate the effect of including or excluding various groups of features, so as not to blindly add features without knowing whether they make sense (cf. Sec. 18.1). Our system should be able to use semantic features (as provided by gazetteers and similar sources), but it should not require them, since semantic resources are typically domain-specific, while our system should be usable for any domains without requiring substantial preparatory effort such as providing suitable sources.

Structural information (e.g. HTML or XML tags) so far has only been used by a few approaches, and typically only by wrapper induction–style approaches such as *Stalker* which do not consider linguistic features and are mainly suited for structured texts. However, the *"Structure matters"* conjecture mentioned in the Introduction (which we will detail in the next chapter) suggests that structural information might be of relevance even for semi-structured and free texts where the usage of linguistic features is generally considered advisable. Hence we will find a way to combine these various sources of information in rich feature representations (cf. Chap. 12). We will also test whether the *"Structure matters"* conjecture actually holds for our evaluation corpora (cf. Sec. 18.1).

As another so far largely unexploited source of information, we will investigate approaches of integrating hierarchical structures of data such as inheritance hierarchies between attributes (cf. Chapters 14 and 20).

7.1.4 Tagging Requirements and Learning Characteristics

In Section 6.4 we have seen that most current IE approaches require a fully annotated set of training texts, while some support *active learning* to reduce the training burden. Active learning reduces the amount of training that a human user has to do, but it still requires a predefined (if unannotated) training set of a sufficiently large size.

However, according to the *"Systems will be used"* assumption we already voiced in the Introduction (and will explain in more detail in the next chapter), the reliance on a predefined fixed-size training set can be a hurdle for many real-life applications: it prevents the system from being used unless a sufficiently large set of training texts has been assembled, and it makes it harder to adapt the system to changes in the corpus.

To address these issues, our system will support *incremental learning* (cf. Sec. 3.3) as an alternative to batch training over fully annotated training corpora. Like active learning, incremental learning reduces the training burden, but additionally it makes it possible to start using the system without a predefined training corpus and it allows allowing successive refinement of an existing extraction model by dynamically adapting it to new training data—the effects of these advantages will be evaluated in Section 18.2. Also, for the user providing the training data incremental learning might been more agreeable than active learning since (s)he stays in control, while in the case of active learning it is the system that decides which documents the user should deal with instead of the other way round.

7.2 Further Requirements

7.2.1 Input/Output Requirements

If we want to be able to handle document structure information, as stated above (Sec. 7.1.3), we cannot just limit our system to handling plain text input as most IE systems do, since in plain text format almost all structural information is lost.[1] Hence our system should also be able to process structured document formats. But there are many such formats, and obviously it would be impossible to support all of them; on the other hand, requiring conversion to one specific structured format would often result in the loss of some structural information which cannot be expressed in the target format.

This problem can be avoided by fixing not a specific structured text format but a "meta-format" that can be used to express almost any structured format. The obvious choice of a meta-format is XML since this generic markup language has already gained widespread acceptance as a meta-format. Accordingly, our system should be able to handle input texts in any XML-based formats in addition to plain text input. Using a meta-format instead of a specific format (such as HTML) means that the *meaning* (semantics) of structural elements is not known in advance, so our system must be able to *learn* the meaning of elements in so far as they are relevant—how to solve this problem will be treated in Sec. 12.2.

[1] At least *explicit* structural information, implicit structural information is another matter—a point we will return to in Sec. 12.1.

In case of plain text input, *answer keys* (user-provided annotations of the expected attribute values for training or for evaluating the system) can be stored inline within the text. This makes it easy to provide answer keys without the need for specific annotations tools, but this is a somewhat brittle solution and it might not work in case of XML input without interference with the document markup. Hence our system should be able process both answer keys provided externally in a database/relational style (for maximum flexibility) as well as inline (for easier use).

While the ultimate goal of schema-based extraction is to store the extracted information in a database that can be queried, just storing the extracted attribute values is not enough. To allow judging the reliability of extracted attribute values, the system should also provide a measure of certainty of its results (a probability estimation of a prediction being correct)—this makes it possible, for example, to only query extracted information whose estimated reliability is beyond a user-defined threshold, or to manually review and correct extractions below a threshold. Both for such manual reviewing and for automatic evaluation is also necessary to provide meta-data that allows anchoring each extracted attribute value in the text it was extracted from.

These issues regarding input and output will be treated in Chapter 9 in more detail.

7.2.2 Architectural Requirements

Current IE system tend to be very tightly coupled and lack a modular architecture, making it hard to exchange parts of an approach or to modify the preprocessing components. Conversely, our system should be designed in a generic way, using a modular architecture that allows modifying and exchanging the various components independently of each other.

The architecture and development API of the system should be clean and well-documented, and the software should be portable to different systems (portability should not be hard to realize, as we will use Java as implementation language).

7.2.3 Evaluation Requirements

Our work will include a detailed evaluation of our approach. For evaluation, we will use two of the most frequently used standard IE corpora. The two corpora should represent very different aspects of the typical range of texts our approach is meant to handle, one representing *semi-structured*, informal texts that are typical for e-mail messages and similar day-to-day communications, and the other representing "classical", fully grammatical *free texts* as can be found in formal sources such as newspapers.

As mentioned above, we will also perform an ablation study to measure the effects of various groups of features on the results (Sec. 18.1) and we will evaluate whether extended feature sets considering type hierarchies can improve results (Chap. 20). We will also investigate the utility of incremental training for reducing the human training effort (Sec. 18.2).

Regarding the architectural modularity (cf. Sec. 7.2.2 above), we will also perform a systematic analysis of switching one core component (Chap. 19). Finally we will

analyze the mistakes made by our system to gain a better insight into weaknesses of our system and general difficulties of information extraction (Chap. 21).

7.3 Chosen Approach

After discussing these aims and requirements we would like to fulfill, we are now ready to chose the kind of approach to pursue in our work.

The first question is whether it should be statistical, rule-based, or knowledge-based. Knowledge-based approaches are really out of the questions since we stated already (in Sec. 7.1.3) that our system should allow, but not *require* the usage of domain-specific semantic information, while for knowledge-based approaches they are the primary source of information.

Various of the aims we have defined suggest choosing a statistical approach instead of a rule-based one. Especially, *incremental training* would be hard to reconcile with a rule-learning approach since extraction rules are generally constructed from a whole set of training texts and cannot be updated afterwards without a full retraining. Also, probability estimation is usually at the core of statistical systems, while most rule-based approaches do not provide a measure of certainty that would allow estimating the reliability of proposed extractions.

Moreover, statistical systems tend to be more robust regarding noise and irregularities in the input, making them specifically suitable for an approach that is meant to handle *semi-structured* in additional to *free texts* (cf. Sec. 7.1.2). Such informal or quickly written texts often lack both linguistic exactness and structural regularity, making it hard for rule-based approaches to learn reliable rules.

A further point that makes us opt for a statistical approach is that the best current statistical IE systems tend to outperform rule-based approaches, as will become apparent during evaluation when we compare our results with those reached by the best other approaches (Chap. 17).

More specifically, we will derive our approach from the family of token-classification approaches (Sec. 4.4) because of the high flexibility it offers. An advantage of token-classification approaches is their being able to handle any feature sets, without generally requiring that features be independent of each other (a requirement that would be very unrealistic in many cases). This makes this family of approaches specifically suitable for use with rich feature sets (cf. Sec. 7.1.3).

Token classification is also a good basis for architectural modularity, as postulated above (Sec. 7.2.2). We will design and implement our system in a way that makes it easy to replace or modify the various core components (classification algorithm, tagging strategies, context representations) independently of one another.

Token-classification approaches are very competitive with other (both statistical and rule-based) approaches, as shown by the fact that both *ELIE* and our own tend to be among the (if not to be the) best systems on each evaluated corpus (cf. Chap. 17 and [Fin06]).

7.4 Non-Goals

To make the scope of this work clearer, it is also helpful to point out which related areas and tasks will *not* be covered in this thesis:

We are not working on methods for *creating or improving target schemas.* For the purpose of this work, target schemas are assumed to be given (cf. Sec. 9.1 for more on the target schemas we will be using). While usually target schemas are created manually (since human users tend to know best what is of interest to them), automatic or semi-automatic procedures for designing or refining them are possible too, but they will not be treated in this work.

This thesis is focused on *supervised* learning, since information extraction is generally modeled as a supervised learning task (as already stated in Sec. 3.1). Hence, our system will require training data (sample texts annotated with answer keys) provided by human users used as target function. We will not consider unsupervised methods that try to work without training data, nor mining algorithms that try to discover potentially relevant facts without an explicit target schemas.

We consider this human-provided training data as a "gold standard" that is not to be judged, so we will not perform any kind of "meta-analysis" of the extracted attribute values, such as trying to discover whether facts expressed in a text are true or reliable, or whether texts are trustworthy or objective.

We are not trying to create a *complete* system for populating databases from textual documents as described in Chap. 3—our system is only meant to be usable as *one core step* of such a system (cf. Sec. 7.1.1). How the remaining steps could be addressed and integrated with our system will be discussed in the "Future Work" section of this thesis (Sec. 22.3).

Also, while Part IV contains evaluations of some of the core parameters, in general we have refrained from performing extensive *parameter variation* tests. Determining optimum parameter values is mainly relevant when tuning for a specific task—we are leaving this for future work, performing parameter evaluations only where new insights can be expected from doing them.

Now we have formulated the aims as well as the scope of our own work, but there remain some issues which we should address before we are ready to introduce and discuss the chosen approach in detail (which will happen in Part III). The need for a more detailed coverage of target schemas and input/output formats has already been pointed out above. Prior to doing so in Chapter 9, it is useful the recapitulate and detail the novel assumptions and conjectures we have made for our algorithm as well as the general assumptions that underlie all IE approaches but are seldom spelled out explicitly. That will be the goal of the next chapter.