8 Evaluation of Information Extraction Technologies

8.1 Introduction

If we build technologies, we would like to evaluate our systems in order to see how they behave with regard to a golden standard and how they compare to other existing technologies for the same task. This is also true for information extraction. Depending on the application, certain performances are measured. For instance, in some cases high *precision* results are of primordial importance, when the extraction results are not manually controlled, while in other cases where the machine extraction is only performing an initial filtering of the information that eventually is manually selected, a high *recall* of the extraction is important. High precision means that the extracted information does not contain any or only very few errors. High recall refers to the situation where all or almost all information to be extracted is actually extracted. An example of the former is extracting the price of an air flight from the World Wide Web. An example of the latter is intelligence gathering, where the analyst wants to find as much as possible valid information on the locations of a certain crime, which afterwards will be manually processed in combination with other evidence. In some tasks errors do not weight equally, as some errors are perceived more severe with regard to the use or further processing of the extracted information. This is, for instance, the case when similar phenomena are grouped and the correctness of the clustering is measured (e.g., in coreference resolution). It is not always easy to define appropriate evaluation measures, especially not in natural language processing tasks. In addition, a different weighting of certain types of errors introduces an element of subjectivity and context-dependency into the evaluation process.

Many of the metrics were already defined during the *Message Understanding Conferences* (*MUC*) in the 1990s. Evaluation is sometimes a complicated and controversial issue. The MUC scoring program and criteria were an important first step in confronting this problem.

The *Automatic Content Extraction* (*ACE*) competition currently develops its own metrics. In the course of the last decades information retrieval has developed several evaluation metrics in the framework of the *Text REtrieval Conferences* (*TREC*) (van Rijsbergen, 1979; Baeza-Yates and Ribeiro-Neto, 1999; Voorhees and Harman, 2005).

Information extraction is usually not a final goal, but assists in other tasks such as information retrieval, summarization or data mining. Many evaluation measures aim at an *intrinsic evaluation*, i.e., the performance of the extraction task is measured. It might be valuable to perform an *extrinsic evaluation*, i.e., measuring the performance of another task in which information extraction is an integral part. In this book the extrinsic evaluation measures focus on measuring the performance of information retrieval in which extraction plays a role.

Most of the evaluation criteria that we discuss regard *qualitative criteria*. They measure the quality of the results. Accuracy is here of most importance, but other measures such as recall and precision cannot be neglected. Once information extraction is applied in information retrieval tasks, and large documents collections are consulted, a high precision is often important. In other situations one can be confronted with incomplete and imperfect relevance information. For these situations specific evaluation metrics are designed. Besides the quality of the results, *other performance measures* that are important in any text processing task, and application specific measures come into play.

8.2 Intrinsic Evaluation of Information Extraction

A first group of evaluation measures concerns the *intrinsic evaluation* of the results of an information extraction task by comparison with some golden standard (Sparck Jones and Galliers, 1996, p. 19 ff.). Information extraction is a classification task. The assigned classes can be compared with the ideal class assignment, which is usually determined by a human expert.

In many information extraction tasks classes can be objectively assigned and there is seldom a discussion about which classes to assign (e.g., named entity recognition, noun phrase coreference resolution). However, there are tasks for which the assignment is less clear cut (e.g., certain semantic roles and classification of modifiers of nouns). In the latter case it is supposed that so-called *inter-annotator agreement* is sufficiently high (e.g., more than 80%). For some tasks human evaluators do not agree on an golden standard. Inter-annotator agreement is usually computed with a reliability

measure, the most common being the α-statistic (Krippendorff, 1980), the κ-statistic (Carletta, 1996) and Kendall's τ-value (Conover, 1980).

It is often difficult to obtain an annotated test set that is large enough to assess the performance of a system. If the performance of different systems is to be ranked, one is tempted to consider only these instances for classification by the human expert on which most systems disagree. These are definitely hard cases. However, the instances on which the systems agree can be completely wrongly classified. Ignoring them in the evaluation can still give a biased impression of absolute performance.

8.2.1 Classical Performance Measures

Information extraction adopts the typical evaluation measures for text classification tasks being recall and precision, their combination into the F-measure, and accuracy.

The effectiveness of automatic assignment of the semantic classes is directly computed by comparing the results of the automatic assignment with the manual assignments by an expert. When classes are not mutually exclusive (i.e., several classes can be assigned to one instance), binary classification decisions are the most appropriate.

Table 8.1 summarizes the relationships between the system classifications and the expert judgments for the class C_i in case of a binary classification (Chinchor, 1992; Lewis, 1995). They form the basis for the computations of recall, precision and the F-measure.

$$R = a / (a + c) \tag{8.1}$$
$$P = a / (a + b) \tag{8.2}$$
$$Fal = b / (b + d) \tag{8.3}$$

Recall (R) is the proportion of class members that the system assigns to the class. *Precision* (P) is the proportion of members assigned to the class that really are class members. *Fallout* (Fal) computes the proportion of incorrect class members given the number of incorrect class members that the system could generate. Ideally, recall and precision are close to 1 and fallout is close to 0.

Table 8.1. Contingency table of classification decisions.

	Expert says yes	Expert says no	
System says yes	a	b	$a + b = k$
System says no	c	d	$c + d = n - k$
	$a + c = r$	$b + d = n - r$	$a + b + c + d = n$

where

n = number of classified objects

k = number of objects classified into the class C_i by the system

r = number of objects classified into the class C_i by the expert.

When comparing two classifiers, it is desirable to have a single measure of effectiveness. The *F-measure*, derived from the E-measure of van Rijsbergen (1979, p. 174 ff.) is a commonly used metric for combining recall and precision values in one metric:

$$F = \frac{(B^2 + 1)PR}{B^2P + R} \tag{8.4}$$

where

P = precision

R = recall

β = a factor that indicates the relative importance of recall and precision, when β equals 1, i.e., recall and precision are of equal importance, the metric is called the harmonic mean (F_1-measure).

Recall errors are referred to as false negatives, while precision errors regard false positives. The *error rate* (*Er*), which is also based on the contingency Table 8.1, takes into account both errors of commission (b) and errors of omission (c).

$$Er = (b + c) / n \tag{8.5}$$

In Table 8.1 it is easy to see that the classical measure of *accuracy* is computed as:

$$\text{Accuracy} = (a + d) / n \tag{8.6}$$

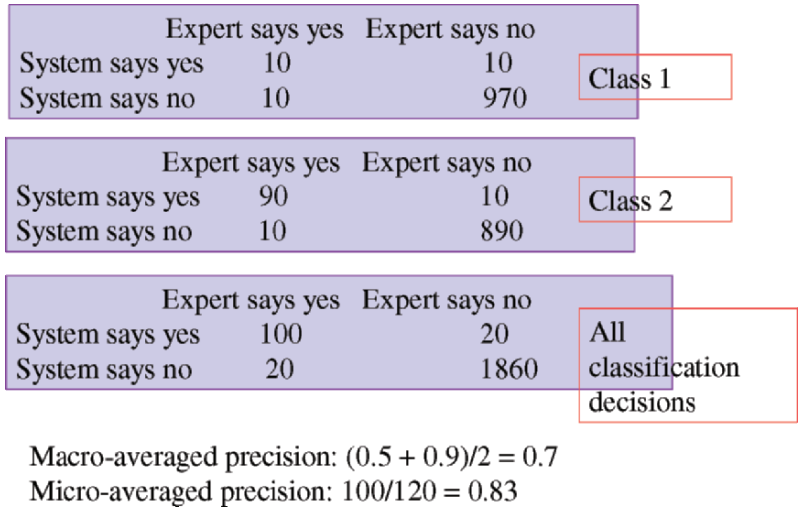

Fig. 8.1. Example of macro-averaged and micro-average precision.

Often multiple classes are assigned (e.g., assigning semantic roles to sentence constituents), pointing to the need for an overall assessment of the performance of the extraction system. In this case the results of the above measurements for each class can be averaged over classes (*macro-averaging)* or over all binary classification decisions (*micro-averaging)* (Fig. 8.1) (Lewis, 1992). The latter way of averaging provokes that categories with many examples have a larger impact upon the results.

In some information extraction tasks, classes are mutually exclusive, i.e., only one class can be assigned to the information constituent. In this case *accuracy* is an efficient performance measure, where accuracy is computed as the proportion of correct assignments to a class in all assignments. It can be seen that in this case micro-averaged precision and micro-averaged recall equal accuracy.

In information extraction both the detection of the information (e.g., detection of the boundary of an entity mention) and the recognition (classification) of a mention should be evaluated. For both tasks usually the same evaluation metrics are used. The result of information extraction is often a probabilistic assignment. None of the above metrics takes the probability of the assignment into consideration.

8.2.2 Alternative Performance Measures

In cases when information is classified by grouping the tokens into clusters, adequate performance measures have been designed that are a variation of the classical recall and precision measures. The metrics are usually illustrated with the task of noun phrase coreference resolution. Building noun phrase coreference chains regards the grouping of noun phrases into clusters. For instance, in the following example **John saw Mary. This girl was beautiful. She wore a red dress** one cluster should contain **Mary**, **girl** and **she** apart from two singleton clusters respectively containing **John** and **dress.**

When evaluating or validating the clustering in information extraction, often the Vilain metric (official metric used in the MUC competition) or the B-cubed metric (Bagga and Baldwin, 1998) is used. In both validations clusters that are manually built by a human expert are compared with the clusters that are automatically built.

The *Vilain algorithm* takes into account the number of links that should be added 1) to the automatic output in order to arrive to the manual clustering and 2) to the manual output in order to arrive to the automatic one. The former number influences the recall measure R, while the latter influences the precision measure P. Formally one defines:

For a cluster S of entities in the manual output, $p(S)$ is a partition of S relative to the automatic response. Each subset of S in the partition is formed by intersecting S and those automatic clusters that overlap S. For example, if one manual cluster is $S = \{A, B, C, D\}$ and the automatic clustering is $\{A,B\}$, $\{C, \ldots\}$, $\{D, \ldots\}$, then $p(S) = \{\{A,B\},\{C\},\{D\}\}$.

$c(S)$ is the minimal number of "correct" links necessary to generate the cluster S.

$$c(S) = (|S| - 1) \tag{8.7}$$

$m(S)$ is the minimal number of "missing" links in the automatic clusters relative to the manual cluster S.

$$m(S) = (|p(S)| - 1) \tag{8.8}$$

The recall error for the manual cluster S is the number of missing links divided by the number of correct links:

$$\frac{m(S)}{c(S)} \tag{8.9}$$

The recall is thus:

$$\frac{c(S)-m(S)}{c(S)} \tag{8.10}$$

This equals:

$$\frac{|S|-|p(S)|}{|S|-1} \tag{8.11}$$

Extending this recall measure to the whole clustering output leads to:

$$R=\frac{\sum_{j=1}^{k}(|S_j|-|p(S_j)|)}{\sum_{i=1}^{k}(|S_j|-1)} \tag{8.12}$$

for each cluster j in the k clusters of the output.

The precision measure is obtained by switching the roles of the automatic and manual clustering, yielding:

$$P=\frac{\sum_{j=1}^{k}(|p(S_j)|-|p(S_j)\cap S_j|)}{\sum_{j=1}^{k}(|p(S_j)|-1)} \tag{8.13}$$

The B-cubed algorithm takes into account the number of entities that should be added 1) to the automatic output in order to arrive to the manual one and 2) to the manual output in order to arrive to the automatic one. The former number influences the recall measure R_i, the latter number

influences the precision measure P_i. Formally, given n objects, we define for each object i:

$$R_i = \frac{co_i}{mo_i} \tag{8.14}$$

$$P_i = \frac{co_i}{ao_i} \tag{8.15}$$

where co_i = number of correct objects in the cluster automatically built that contains object i
mo_i = number of objects in the cluster manually built that contains object i
ao_i = number of objects in the cluster automatically built that contains object i

The final recall R and precision P that consider all n objects of the clustering are respectively computed as follows:

$$R = \sum_{i=1}^{n} w_i \cdot R_i \tag{8.16}$$

$$P = \sum_{i=1}^{n} w_i \cdot P_i \tag{8.17}$$

where w_i are weights that indicate the relative importance of each object (e.g., in noun phrase coreference resolution the pronoun i could be weighted differently than the noun i). All w_i should sum to one and they are often chosen as $1/n$.

Both the Vilain and B-Cubed metrics incorporate some form of subjectivity in measuring the validity of the clusters. The Vilain metric focuses on "What do I need to do in order to get the correct result? ", and not in terms of "Is the result that the system obtains correct or not". The Vilain algorithm only rewards objects that are involved in some relationship. Determining that the object is not part of a cluster with another object is unrewarded. In this classic Vilain metric, all objects are treated similarly. In the B-Cubed algorithm, an object's relationship with all other objects in its cluster can be weighted by a weighting parameter.

In analogy with the above measures, one can design other approaches for cluster validation, for instance by taking into account the number of wrong entities in one cluster.

8.2.3 Measuring the Performance of Complex Extractions

An information extraction task is often composed of different recognition tasks, hence the idea of using one evaluation score that evaluates different recognitions. Such a score is valuable when detecting complex content, e.g., content characterized by relations between content elements. Evaluation scores that measure the performance of complex extractions have been designed during the ACE competition (ACE 2005).

The metrics used by the ACE competition compute a value score *Value* for a system defined by the sum of the values of all of the system's output entity tokens, normalized by the sum of the values of all reference entity tokens, i.e., the sum of the ideal score of each token that should be recognized. The maximum possible *Value* score is 100%.

$$Value = \frac{\sum_i Value(sys_i)}{\sum_j Value(ref_j)} \tag{8.18}$$

where

sys_i = value of each system token i based on its attributes and how well it matches its corresponding reference token

ref_j = value of a reference token j.

The tokens are the information elements recognized by the system. The value of a system token is defined as the product of two factors. One factor represents the inherent value of the token, the other assesses how accurately the token's attributes are recognized or the token's mentions are detected. In other words, it is evaluated whether content (e.g., an entity relation, a timex), its attributes and its arguments are recognized correctly. For instance, in a relation recognition task the arguments are the entities that form the relation.

There are two ways to look at content. One way reflects the linking of similar content which is referenced within and across documents where this content (e.g., entity, relation) receives a unique identification number.

The evaluation includes the recognition of the different mentions of that content. A second way is to consider the recognition of each content element and its attributes independently.

We will focus here on the first type of evaluation, because it is the most relevant for complex recognition tasks.

$$Value(sys) = ElementValue(sys) \cdot ArgumentsValue(\{Arguments(sys)\}) \tag{8.19}$$

where *sys* is the content element considered (e.g., *sys* can be an entity, a relation, an event, etc.). *sys* can refer to a system token or a reference token. *ElementValue*(*sys*) is a function of the attributes of the element and, if mapped to the reference element, it judges how well the attributes match those of the corresponding reference element. The function can be defined according to the type of content element that is evaluated. For instance, in a named entity recognition task the inherent value of an entity element is defined as the product of the token's attribute value parameters and of its attribute types (e.g., the characteristics of the entity and the type of entity). This inherent value is reduced for any attribute errors (i.e., for any differences between the values of the system and the reference attributes) using error weighting parameters, $\{W_{err\text{-}attribute}\}$. If a system token is unmapped, then the value of that token is weighted by a false alarm penalty, $W_{E\text{-}FA.}$

The second factor in Eq. (8.19) determines how accurate the information element's mentions or arguments are detected. The detection of mentions refers to the detection of arguments in an equivalence relation between different mentions (e.g., the correct resolution of coreferring content elements). In other types of relations other arguments can be detected, such as the recognition of the arguments of an action or a speech act, or the recognition of the necessary parts of a script.

The exact function for the computation of the element value and the mentions value depends on the extraction task and on what aspects of its performance that are considered important for a certain application. The functions are here illustrated with the example of the recognition and normalization of temporal expressions in text.

$$Value(sys) = ElementValue(sys) \cdot MentionsValue(sys) \tag{8.20}$$

The *ElementValue* (*sys*) here depends on how well the attributes of the system token *sys* mach those of the corresponding reference token. The intrinsic value of a timex token is defined as a sum of attribute value parameters, *AttrValue*, summed over all attributes $a \in A$ which exist and which are the

same for both the system and reference tokens. In the recognition and normalization of temporal expressions *A* is composed of the following attributes. Temporal expressions to be recognized include both absolute expressions and relative expressions (*Type*). In addition, the attributes include the normalized time expression (*Val*) (e.g., **2005-9-23**), the normalized time expression modifier (*Mod*) (e.g., **approximate**), a normalized time reference point (*AnchorVal*) (e.g., **2005-9-5)**, a normalized time directionality *(AnchorDir*) (e.g., **before**), and a flag that ascertains that *Val* is composed of a set of time expressions (*Set*). These attributes follow the conventions of the "TIDES 2005 standard for annotations of temporal expressions". If a system token is unmapped, *ElementValue* (*sys*) is zero.

$$ElementValue(sys) = \sum_{a \in A} \begin{Bmatrix} AttrValue(a) & \text{if } a(sys) = a(ref) \text{ and } sys \text{ is mapped} \\ 0 & \text{otherwise} \end{Bmatrix} \tag{8.21}$$

MentionsValue (*sys*) is simply the sum of the mention values (*MMV*) of a system token. A mention's *MMV* is simply 1, if the system token's mention maps the corresponding reference token. If the system token's mention is unmapped, then the *MMV* is weighted by a false alarm penalty factor, $W_{M\text{-}FA}$ and also by a coreference weighting factor $W_{M\text{-}CR}$. The latter refers to the penalty when the system mention happens to correspond to a legitimate reference mention, but one that does not belong to the corresponding reference token. For each pairing of a system token and a reference token, an optimum correspondence between system mentions and reference mentions that maximizes the sum of *MMV* over all system mentions is determined and used, subject to the constraint of a one-to-one mapping between system and reference mentions.

$$MMV(mention_{sys}) = \begin{Bmatrix} 1 & \text{if } mention_{sys} \text{ is mapped} \\ -(W_{M-FA} \cdot W_{M-CR}) & \text{otherwise} \end{Bmatrix} \tag{8.22}$$

$$MentionsValues(sys) = \sum_{\text{all docs}} \left(\sum_{\text{all sys mentions in doc}} MMV(mention_{sys}) \right) \tag{8.23}$$

Table 8.2. Examples of values of weight parameters used in the attribute matching of the recognition and normalization of temporal expressions.

ElementValue parameters						
Attribute	*Type*	*Val*	*Mod*	*AnchorVal*	*AnchorDir*	*Set*
AttrValue	0.10	1	0.10	0.50	0.25	0.10
$W_{E\text{-}FA} = 0.75$						
MentionsValue parameters						
$W_{M\text{-}FA} = 0.75$		$W_{M\text{-}CR} = 0.00$		*MinOverlap* = 0.30		

System mentions and reference mentions are permitted to correspond only if their extents have a mutual overlap of at least *MinOverlap*. In the frame of ACE 2005 overlap is simply defined as the normalized number of characters that are shared by the two strings.

From the above it is clear that several parameters have to be a priori set. In the ACE 2005 competition these parameters were set as shown in Table 8.2. In order to obtain a global evaluation of a system's performance in temporal expression recognition and normalization in text, a final score is computed according to Eq. (8.18). This score is 100% when all timexes, their attributes and mentions are perfectly recognized and normalized.

The mutual overlap parameter determines the conditions under which the two mentions are allowed to map. In MUC-4 (1997) a partial matching of mentions was allowed. In case of a partial matching, the performance score is decreased by a predefined factor. Lee et al. (2004) propose to measure the performance of the recognition according to each boundary condition of strict, left, right and sloppy: Strict means that the boundaries of the system and those of the answer match on both sides, left means that only the left boundary of the system and that of the answer match, right means that only the right boundary of the system and that of the answer match, and sloppy means that the boundaries of the system and those of the answer overlap.

Evalation of several subtasks and integrating the evaluation score in one metric often demands weighting of the subscores based on a priori defined parameters. This is illustrated with the performance measure discussed in this section. Such an approach is subjectively colored by the many parameters that have to be tuned. But, the metric makes it clear that there is an absolute need to evaluate a combination of extraction tasks. In the future this demand will only increase as the different extraction tasks will eventually lead to the understanding of texts.

8.3 Extrinsic Evaluation of Information Extraction in Retrieval

Classical evaluation in information retrieval relies on *recall* and *precision* values (possibly combined in a *F-measure*) to assess the performance of the retrieval. We refer to Eqs. (8.1), (8.2) and (8.5) where the binary class considered is now the one of relevancy or correctness of the answer in the result or answer list returned by the information retrieval system. Formally, we define *recall* (*R*) and *precision* (*P*) respectively as:

$$R = \frac{ard}{trd} \tag{8.24}$$

$$P = \frac{ard}{ad} \tag{8.25}$$

where *ard* = number of relevant documents in the result list
trd = total number of relevant documents in the document base
ad = number of documents in the result list.

Note that the term "documents" is interpreted here very broadly and encompasses document elements or passages, sentences or phrases, apart from regular documents.

Currently, some measures take into account the ranking, which is the relative ordering of the retrieved documents by perceived relevance. One of the earliest metrics is the *reciprocal answer rank* (*RAR*) developed for evaluating the performance of question answering systems and whose weights influence the correctness of the answer according to its position in the answer list, while decreasing the influence of an answer further down in this list.

Another metric is the *mean average precision* (MAP), also referred to as the mean non-interpolated average precision (Buckley and Voorhees, 2002) computed as a mean over a set of queries. The average precision (AP) is computed after every retrieved relevant document, using zero as precision for relevant documents that are not retrieved, and then averaged over the total number of retrieved relevant documents for a query.

Suppose we have *trd* relevant documents for a given query in our test collection, AP is defined as:

$$AP = \frac{1}{trd} \sum_{r=1}^{trd} P_r \tag{8.26}$$

$$P_r = \frac{ard_r}{r} \tag{8.27}$$

where ard_r = the number of relevant documents in the result list up to the position of the r^{th} relevant document. If the r^{th} relevant document does not occur in the result list, $P_r = 0$.

A reader looses his or her time when looking at non-relevant documents that are ranked higher than relevant documents. So, a good result list should have as few as possible non-relevant documents ranked higher than relevant documents. This is, for instance, reflected in the *bpref* (*binary preference*) measure, which measures the number of faulty orderings in the result list, i.e., orderings where a non-relevant document is ranked before a relevant document (De Beer and Moens 2006).

$$bpref = \frac{1}{ard} \sum_{r=1}^{ard} (1 - \frac{nn_r}{nn}) \tag{8.28}$$

where nn_r = the number of non-relevant documents in the result list up to the position of the r^{th} relevant document and nn = the number of non-relevant document in the result list.

The soundness of a variant of this matric and its robustness in the face of incomplete and imperfect relevance information are discussed and demonstrated by Buckley and Voorhees (2004). By incomplete judgments we mean that the result list does not contain all the relevant documents. An imperfect judgment refers to a situation in which a document of the result list is no longer part of the document collection. Both situations occur in current search settings.

One document or answer might be more relevant than another one in the list of retrieved documents. It is our conviction that for many applications, binary relevance judgments are rarely adequate to fully express the perceived relevance level experienced by end users. Relevance should be considered a fuzzy variable, as it is - besides other factors - largely dependent on the utility of the judged documents for satisfying the user's (underspecified) information needs. Therefore, De Beer and Moens (2006) have proposed a generalization of the *bpref* measure that measures the intrusion of less relevant documents before and between more relevant documents.

We are not aware of any metric that measures the performance of information extraction and information retrieval in a combined way. Such a metric could be useful to compare retrieval systems that use different information extraction technologies.

In information retrieval there is a growing need for evaluation metrics that judge answers to information questions. The answers are extracted from a document or even synthesized from different document sources (see Chap. 10). In such a setting it is important that the answer is complete and correct, i.e., it contains all and only correct elements and the elements are connected with the right relationships. Research into evaluation metrics for text summarization might be adopted. Such metrics are currently under development in the community of the Document Understanding Conference (DUC). When different answers are retrieved from the document collection, e.g., when the query or information question could be interpreted in different ways, the evaluation metric should also assess that the most relevant answers come first, or are preceded by only very few non-relevant or less relevant answers.

8.4 Other Evaluation Criteria

When dealing with text, other criteria for judging the performance of information extraction systems are important. Evaluating natural language text is extensively discussed in Sparck Jones and Galliers (1997).

A first evaluation criterion regards the *computational complexity* of the information extraction and of the storage overhead. Even if computer power has dramatically grown, extracting content from texts is computationally expensive and care should be taken to use efficient computations whenever possible. When information extraction results are added to document indices in retrieval systems, a balance should be sought between the number of allowable computations at query time and the storage overhead caused by intermediary results that were a priori calculated. It could be measured how large the indexing overhead is and how this effects the retrieval performance for certain applications.

Another concern is *linguistic coverage.* Although becoming a smaller problem over the years, some types of linguistic phenomena cannot yet be covered in a certain language as the necessary technology and resources are not yet developed. Or the linguistic tools might not yield sufficient reliance in terms of qualitative performance. This situation constraints certain information extraction tasks (e.g., entity relations recognition relies on a syntactic parse of a sentence). So, when judging the extraction systems,

the evaluation report preferably includes the natural language processing resources and tools that are used, and evaluates their performance for the task at hand. If a part-of-speech tagger or a sentence parser is used, the accuracy of the results can be measured (van Halteren, 1999).

Some information extraction systems might perform well in a limited domain where enough annotated examples are provided to cover all phenomena (all variant linguistic expressions are annotated) and the ambiguity of the language is more restricted. In order to measure the *domain coverage*, the concept of domain has to be specified. This is often difficult. A domain is sometimes associated with a sublanguage. Such a sublanguage is more restricted in its linguistic properties (vocabulary, syntax, semantics and discourse organization) (Grishman and Kittredge, 1986). Typical sublanguage texts may be weather reports and medical discharge summaries of patients. Information extraction from sublanguage domains is thought to be easy. However, linguistic expressions from the standard language or from neighboring domains possibly enter the sublanguage without going through a process of setting up conventions. With regard to information extraction, this means that part of the extraction tasks can be defined across domains and others are very domain specific. As we will see in Chap. 9, some information tasks are much more difficult than others and the degree of difficulty may vary from domain to domain. Rather than considering domain coverage as the proportion of the domain that is covered by the extraction system, it makes more sense to measure the performance of the different extraction tasks.

The information to be extracted is described by the classification scheme or extraction ontology and in order to have comparable performance measures, this classification scheme should be accepted by a large community. This brings us to the problem of standardization. The output of the extraction system (i.e., the semantic labels) should be as much as possible standardized, so as to ensure *interoperability* and *comparability of systems* and to facilitate that the output can be processed by other systems such as retrieval, data mining and summarization tools.

Another performance criterion is measuring the *extensibility of the extraction system*. A system can be extended in two ways: By enlarging the feature space or by enlarging the extraction scheme or ontology. Enlarging the feature space often regards inclusion of extra linguistic phenomena because of the availability or advancement of natural language processing resources. The second enlargement regards the domain coverage, i.e., the classification scheme is extended in order to cover extra intra-domain or inter-domain classes. Extensibility is difficult to quantitatively measure. However, one could note the differences in performance after the

system is extended. Also, the necessary changes to the system in this adaptation should be described in any performance report.

Related to extensibility is *portability*, i.e., the capability of a system to be transferred from a language or subject domain to another one and the amount of extra work (e.g., in the form of drafting knowledge rules or of annotating and training a system).

We can also measure how much *time* it takes *to train* an extraction system, whether it is a system that is built from scratch or whether the system is extended or ported to another language or domain. The time to train a system largely depends on the size of the classification scheme used and on the complexity of the examples that are associated with certain classes. This is also difficult to quantitatively measure. First of all, there is the cost of annotation. Even with sophisticated annotation tools which have to be adapted to changing features and classes, annotation is a real burden, which one wants to reduce as much as possible. Another question to be asked is: Can the system be trained incrementally without starting from scratch when new labeled examples are available, or when classes or features are updated?

The criteria of extensibility, portability and time to train a system regard the maintenance of the system.

Very often the circumstances in which a system is trained or operates are not ideal. For instance, the *linguistic quality of the input* can be distorted by spelling and grammatical errors (e.g., spam messages). Then, it is definitely worth measuring how robust the system is. The performance can also be compared when all the settings for the extraction are kept constant and the noisy text is replaced by its non-noisy variant.

Finally, there are a number of criteria that are common for many information systems. They regard – among others – *latency* (speed of generating the answer) and *efficient usage of system resources* (working memory, storage, bandwidth in case of distributed information), *scalability to large document collections*, *huge classification schemes,* and *a large number of languages.*

8.5 Conclusions

In this chapter we have given an overview of a number of classical evaluation metrics for assessing the performance of information extraction and information retrieval systems. There is still room for the development of evaluation metrics that measure the quality of the results of retrieval systems that incorporate extraction technology, for instance, when measuring

the completeness and correctness of an answer to an information question by splitting the answer into information elements and their relationships. Because information extraction from text and information retrieval technology that relies on these extraction technologies employ natural language processing tools, performance measures that are commonly applied in human language technology seem useful.

8.6 Bibliography

ACE (2005). The ACE (2005) evaluation plan. Evaluation of the detection and recognition of ACE entities, values, temporal expressions, relations and events.

Baeza-Yates, Ricardo and Berthier Ribeiro-Neto (1999). *Modern Information Retrieval*. New York: Addison-Wesley.

Bagga, Amit and Breck Baldwin (1998). Algorithms for scoring coreference chains. In *Proceedings of the Linguistic Coreference Workshop at the First International Conference on Language Resources and Evaluation* (*LREC'98*) (pp. 563-566). LREC.

Buckley, Chris and Ellen M. Voorhees (2002). Evaluating evaluation measure stability. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 33-40). New York: ACM.

Buckley Chris and Ellen M. Voorhees (2004). Retrieval evaluation with incomplete information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Information Retrieval* (pp. 25-32). New York: ACM.

Carletta, Jean (1996). Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22 (2), 249-254.

Chinchor, Nancy (1992). MUC-4 Evaluation metrics. In *Proceedings of the Fourth Message Understanding Conference* (*MUC-4*) (pp. 22-50). San Mateo, CA: Morgan Kaufmann.

Conover, William J. (1980). *Practical Non-Practical Statistics,* 2nd edition. Redwood City, CA: Addison-Wesley.

De Beer, Jan and Marie-Francine Moens (2006). Rpref - A Generalization of Bpref towards Graded Relevance Judgments. In *Proceedings of the Twenty-Ninth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Grishman, Ralph and Richard Kittredge (1986). *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*. Hillsdale, NJ: Lawrence Erlbaum.

Krippendorff, Klaus (1980). *Computing Krippendorff's Alpha-Reliability*. Thousand Oaks, CA: Sage Publications.

Lewis, David D. (1992). Evaluating and optimizing autonomous text classification systems. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 246-254). New York: ACM.

Lewis, David D. (1995). Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th Annual International ACM SIGIR Conference*

on Research and Development in Information Retrieval (pp. 246-254). New York: ACM.

Sparck Jones, Karen and Julia R. Galliers (1996). *Evaluating Natural Language Processing: An Analysis and Review.* Springer: New York.

Van Halteren, Hans (1999). Performance of taggers. In Hans van Halteren (Ed.) *Syntactic Wordclass Tagging* (pp. 81-94). Dordrecht: Kluwer Academic Publishing.

Van Rijsbergen, Cornelis J. (1979). *Information Retrieval*, 2nd ed. London: Butterworths.

Voorhees, Ellen and Donna K. Harman (Eds.) (2005). *TREC: Experiment and Evaluation in Information Retrieval.* Cambridge, MA: The MIT Press.