9 Learning Algorithm and Application

While in the previous chapters we focused on single steps of induction process of extraction rules, this chapter connects them drawing a complete picture of the learning algorithm. Starting from the pool of correct rules during one learning cycle extraction rules pass the generalization step and come in the pool of induced rules. At this stage the issue, what rules should be generalized, especially what pairs of rules should be merged, is crucial for the quality of induced rules. This chapter introduces a concept of the abstraction degree of a rule that allows to perform controlled generalization avoiding too fast induction and resulting overgeneralization.

The second part of the chapter treats the reverse branch of the learning cycle leading from the induced to the correct rules. Validating the induced rules there is a tradeoff between approving only the best rules at the expense of the overall recall and rejecting only the most unreliable rules taking lower precision into account. To resolve this tradeoff we propose the evaluation of the rule performance based on the precision values achieved by a single rule for every extracted attribute and in total. Optimization of precision thresholds on the training corpus enables an adequate selection of a set of correct rules.

We conclude the chapter with the discussion of appropriate termination criteria for the learning algorithm and the differences between the evaluation and application mode of GROPUS.

9.1 Selection of Extraction Rules for Generalization

Beginning with the initial rules the set of correct rules is extended by the validated induced rules after every iteration of the learning cycle. Although the rules validated in the previous learning cycles are less general, they are not removed from the set of correct rules for two reasons. The fact that these rules have been validated indicates their reliability. Thus even though they may not find as many relevant facts in the test corpus as more general rules added later, their extractions will not reduce the overall performance and most probably contribute to better results (in case that they produce unique correct extractions not also covered by more general rules). Furthermore less general rules can in turn function

as building blocks for rule generalization in any following iteration step.

The set of correct rules contains a mixture of rules with different degrees of generality. Therefore numerous combinations of different rules are possible for rule merging. However, it is not reasonable to generate a general rule from any combination of correct rules. Given that $n * (n-1)$ pairs of n rules can be built, the number of induced rules will grow quadratically per iteration. Besides that a tremendous effort will be required to validate these rules, many of them will not be capable of identification and extraction of information. As we have already argued in sec. 7.2, the best generalization can be achieved merging similar rules since the evidence of characteristic features is enforced. Assessment of rule similarity is therefore used to reduce the range of candidates for rule merging.

Rule similarity is not the only aspect that matters in context of rule merging. The speed of generalization process (i.e. in how many iterations the maximum generalization is achieved) depends on the generality of merged rules. If for example, rules with low generality are merged, the generalization advances very slow while in case of high generality the maximum generalization is achieved in a few steps. While a slow generalization process implies long runtime and many redundant extraction rules, during the fast generalization many reliable intermediary levels of generality may be skipped resulting in overgeneralization and lower precision and recall values. Therefore in the optimal case the generality of rules will be constantly moderately growing so that any rule generated in the iteration i is slightly more general than any rule created during the iteration $i-1$. To guarantee a steady increase of the generality of the set of correct rules and provide its quantitative measure we introduce the concept of *abstraction degree* of a rule, which formalizes the notion of rule generality.

9.1.1 Rule Inclusion and Abstraction Degree

A general rule is supposed to be able to extract relevant information from different texts covering various expression possibilities. Hence the rule generality manifests itself in the number of extracted fragments from a given text corpus, that is, how often a rule can be applied in the training corpus. The quantitative measure of the rule generality can therefore be based on the training examples correctly identified by a rule in the training corpus.

Let $\tau = \{s_1, \ldots, s_n\}$ be the training corpus. Here we can abstract from single texts and regard the training corpus as a set of sentences comprised by its texts. Let $R = \{r_1, \ldots, r_n\}$ be the set of extraction rules. The function $match :: R \times \tau \rightarrow \{true \mid false\}$ determines whether a rule extracts attribute values from a sentence, i.e. whether the rule pattern of the first argument matches the sentences passed as the second argument.

$$\begin{aligned} match(r_i, s_j) = true &\quad \Leftrightarrow \quad r_i \text{ extracts fragments from } s_j \\ match(r_i, s_j) = false &\quad\quad\quad \text{otherwise} \end{aligned}$$

Given a rule and a text corpus the function $global\_match :: R \times \{\tau\} \rightarrow \tau$ calculates the set of sentences of the text corpus from that fragments are extracted by the rule.

$$global\_match(r_i, \tau) = \{s_k \mid match(r_i, s_k) = true\}$$

A rule derived by merging from two less general rules is supposed to inherit and extend their properties so that at least any fact extracted by the two original rules is also extracted by the new rule.In context of generalization the notion of

rule inclusion is very useful describing the relation between more and less general rules. A rule r_i includes the rule r_j if it matches at least the sentences matched by r_j:

$$r_i \supseteq r_j \Leftrightarrow global\_match(r_j, \tau) \subseteq global\_match(r_i, \tau) \tag{9.1}$$

Using the rule inclusion we can formally characterize the inheritance of the properties of the merged rules by the resulting rule. Any rule that is a result of the application of the *merge* function ought to fulfil the following relation:

$$global\_match(merge(r_i, r_j), \tau) \supseteq global\_match(r_i, \tau) \cup global\_match(r_j, \tau)$$

and as a consequence

$$merge(r_i, r_j) \supseteq r_i \wedge merge(r_i, r_j) \supseteq r_j \tag{9.2}$$

After having introduced the concept of rule inclusion we can derive the quantitative measure for rule generality – abstraction degree – as a function $A\_degree :: Rules \rightarrow N$. Initial rules usually can only match and extract information from the training example they were generated from and are therefore the least general rules. Hence initial rules have the minimum abstraction degree that is set to 1. Considering that the generality of a rule depends on the number of the correctly extracted training examples (see above), the abstraction degree of a rule can be defined as the number of initial rules included by this rule:

$$A\_degree(r) = |\{r_k \mid r \supseteq r_k \wedge A\_degree(r_k) = 1\}| \tag{9.3}$$

A rule with an abstraction degree n is therefore supposed to extract at least n training examples in the training corpus. Since every extraction in the training corpus is covered by an initial rule, abstraction degree gives a pretty precise information about the ability of a rule to match different fragments.

To avoid redundancy in the extraction rules we can formulate another constraint for the *merge* function:

$$\begin{aligned} A\_degree(merge(r_i, r_j) \geq A\_degree(r_i) + A\_degree(r_j) \\ :\{r_k \mid r_i \supseteq r_k\} \cap \{r_l \mid r_j \supseteq r_l\} = \oslash \end{aligned} \tag{9.4}$$

Requiring that the abstraction degree of the rule resulting from the rule merging is greater or equal to the sum of abstraction degrees of the merged rules we impose two important conditions on rule merging. One of them states that only rules that include disjoint sets of initial rules can be merged. Merging rules that include identical initial rules would lead to many redundant rules that are very similar to each other or even identical and slow down the generalization process. Another implication is that no extraction should be lost during the rule merging, i.e. the new rule extracts at least. If the rule resulting from merging did not extract all examples extracted by the merged rules, its abstraction degree would be smaller than the sum of abstraction degrees of both merged rules.

9.1.2 Controlling the Rule Generalization

Selecting candidates from the set of correct rules for generalization we are interested in a steady and smooth generality increase of induced rules. Relying on abstraction degree as a reliable and precise measure of rule generality we can control the level of generality of induced rules. To achieve a smooth increase

of rule generality the abstraction degree of induced rules is incremented by 1 in every iteration of the learning cycle. And to induce a rule with a given abstraction degree we can utilize the inequation (9.4) selecting rules with appropriate abstraction degrees for merging from the set of correct rules.

The induction of new extraction rules in the i-th iteration step is performed as follows:

1. Pairs of correct rules (r_k, r_l) with $A\_degree(r_k) + A\_degree(r_l) = i + 1$ are determined and added to the set I.
2. All pairs (r_i, r_j) with $\{r_k \mid r_i \supseteq r_k\} \cap \{r_l \mid r_j \supseteq r_l\} \neq \oslash$ are removed from I.
3. The remaining rule pairs in I are sorted according to the similarity value of the rules $RuleSim(r_i, r_j)$ (s. (7.2))
4. The set M is supposed to contain the rule pairs that will be merged. Iterating over the sorted rules pairs the respective pair with the most similar rules is removed (r_i, r_j). If both members of the pair are not represented in M at least two times – $|\{(r_i, r_k)|(r_i, r_k) \in M\}| < 2 \vee |\{(r_j, r_l)|(r_j, r_l) \in M\}| < 2$ – the pair (r_i, r_j) is added to M. Rule pairs are added to M until the similarity value of the current top pair $RuleSim(r_i, r_j) < similarity\_threshold$.
5. Every rule pair from M is merged and resulting rules are added to the set of induced rules.
6. Every induced rule is applied to the training corpus. If any subset of initial rules is included by more than one induced rule, the best rule is chosen based on the extraction results, while other rules are discarded.

The first step of the algorithm ensures that the induced rules will be more general than the current correct rules having the next higher abstraction degree. In the second step the rule pairs that do not satisfy the condition (9.4) are filtered to avoid validation of redundant rules. Since the similarity of rules increases the evidence of the relevance of common rule features, most similar rules are merged at first. However, merging only very similar rules the typical information expression forms in the training corpus will be overemphasized. Assuring that every rule in the set I is merged at least two times we prevent data sparseness, since merging less similar rules can result in a rule that covers not as many but less common, unusually expressed instances of relevant information than a merger of two similar rules. Sometimes though a reasonable generalization is not possible because rules have too few common properties. Such rule pairs are filtered by the similarity threshold. To avoid redundancy in the set of correct rules, rules that include the same initial rules, that is, produce the same extractions, are removed in the last step of the algorithm except for the rule making the least number of incorrect extractions.

The sample generalization of four initial rules is displayed in fig. 9.1. R_{ij} indicates that the rule includes the initial rules R_i and R_j. Although in this example the abstraction degree of new rules added to the set of correct rules in the iteration i is always $i + 1$, the abstraction degree of induced rules may have a higher value. For instance, the merger of rule R_{14} and R_2 may also cover the extractions of R_3 having an abstraction degree 4. In fact, if the merger includes more rules than the merged rules, it is the evidence of successful generalization, since it extracts new attribute values that the merged rules could not identify.

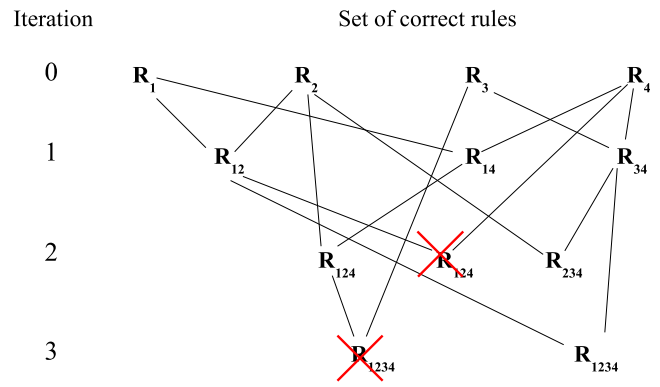

Figure 9.1: Control of generalization of correct rules by abstraction degree

In the iterations 2 and 3 respectively two rules are generated that include the same initial rules: R_{124} and R_{1234}. Even though the pairs of merged rules are restricted by rule similarity, disjoint sets of included rules and constraints concerning abstraction degree, induction of rules that include the same initial rules is quite probable. Theoretically there are $2^{(n-1)}$ possibilities to induce a rule $R_{1...n}$ since merging any dual partition of the set $\{R_1 \dots R_n\}$ leads to $R_{1...n}$ and there are $\frac{|\wp(\{R_1 ... R_n\})|}{2}$ dual partitions. Certainly, most of rule pairs corresponding to the dual partitions will fail to pass the constraints mentioned above and the learning algorithm will terminate for relatively small values of n. But in case of several rules including identical initial rules only one rule has to be selected.

9.1.3 Runtime of the Rule Induction

The runtime of the algorithm essentially depends on the number of correct and induced rules. Suppose there are n initial rules. Since only rule pairs are nominated for merging in that at least one member has been nominated less than two times, there can be $2 * n$ nominated pairs and hence induced rules in the worst case in the first iteration. Suppose that in the worst case all induced rules are validated, contain no redundancy and are all added to the set of correct rules. Thus, at the beginning of the second iteration the set of correct rules contains $n + 2n = 3n$ rules. Since we can use all rules for inducing rules with abstraction degree 3, there are analogously $2*3n$ pairs nominated for merging resulting in $6n$ induced rules and in the worst case in $3n+6n = 9n$ correct rules at the beginning of the third iteration.

Since the number of induced rules is in the worst case twice as large as the number of correct rules, the number of correct rules is tripled at the end of an iteration if all induced rules pass the validation step in the worst case. Therefore we can assess the number of correct rules at the end of iteration i in the worst case as $3^i n$. Assuming that the rule learning algorithm terminates after several constant number of iterations k the number of extraction rules lies in $O(3^k n)$, which is linear to the number of initial rules.

However, the worst case assumptions will hardly hold in all iterations. The number of nominated pairs can, for example, fluctuate between n and $2n$. Some of pairs may not surpass the similarity threshold. There are only $\binom{n}{i}$ subsets of initial rules corresponding to the abstraction degree i so that many redundant rules including the same subset are rejected. And during the validation many induced rules will be discarded because of insufficient precision. Therefore the function derived for the worst case is mainly interesting because it confirms the

linear dependence of the number of extraction rules on the number of initial rules, while the real runtime is significantly lower.

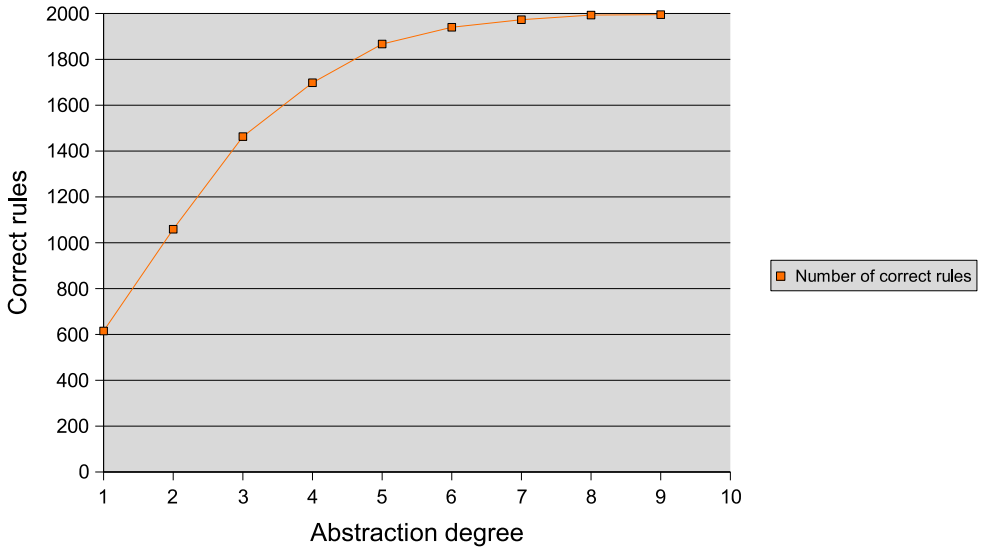

Figure 9.2: Growth of the number of correct rules at the training stage on the MUC corpus

Fig. 9.2 depicts the behavior of the set of correct rules under real conditions during training on the MUC corpus (s. 10.1.3). The initial linear growth continuously slows down for higher abstraction degrees. The number of correct rules is controlled by both the induction algorithm described on p. 105 and validation of induced rules (s. sec. 9.2). Therefore the worst case scenario is hardly relevant for real training as the convergent function of the size of the correct rule set demonstrates.

9.1.4 Utilization of Rule Abstraction and Substitution

Among the presented generalizing heuristics rule merging is supposed to provide the best generalization since it relies on evidence shared by several linguistic patterns that are derived from different texts or text parts. Therefore the merging heuristic plays a major role in the rule learning algorithm being the main instrument for rule induction. However, there are situations when the rule merging cannot be effectively applied.

If the training corpus contains only few training examples for certain attributes, a corresponding small number of initial rules for these attributes will be generated. Among few rules it is difficult to find rule pairs that are similar enough to perform an effective rule merging. At least some of initial rules can not be generalized by rule merging and are basically ignored. Since these rules may contain valuable extraction patterns and contribute to the more comprehensive extraction of such underrepresented attributes. To exploit the capabilities of such ignored initial rules, abstraction heuristic is applied for their generalization.

If a correct rule is not represented at least two times in the set M that contains rule pairs to be merged (refer to p. 105), it will be abstracted. Analogously to the merged rules the abstracted rule is added to the set of induced rules and passes the rest of the learning cycle. Abstraction degree of rules generalized by the abstraction heuristic corresponds as in case of merged rules to the number of included initial rules. An abstracted rule can therefore participate in rule merging in the next iteration.

After several iterations any attempt to generate a more general rule merging any two correct rules fails because the resulting rule produces too many incorrect extractions and cannot be validated. At this point rules have achieved their maximum abstraction degree and rule merging cannot contribute to the improvement of extraction results any more. Because of heterogeneous texts, small number of training examples the set of correct rules may still have a low coverage of relevant information failing to extract many attribute values. At this stage of the induction process the substitution heuristic can be leveraged to increase the recall of extraction rules.

The correct rules with high abstraction degrees induced in the last iterations of the learning cycle contain both general encoding of the extracted parts and generalized context representation. Thus substitution of the encodings of attribute values in such a rule by the corresponding encodings of another rules results in a new general extraction rule. This new rule features unprecedented combinations of extracted information and its context and may therefore identify new facts that the rules derived by rule merging failed to identify and increase the overall recall. Since the rules induced by the substitution heuristic comprise general parts, no further generalization is pursued and they are added to the set of correct rules after the validation step.

9.2 Validation of Induced Rules

9.2.1 The Purpose of Validation

Extraction rules induced by the three generalizing heuristics are not perfect. In the early stages of the learning algorithm they do not sufficiently abstract from the training examples they are derived from and suffer from low covering potential. While in the later iterations rules achieve sufficient abstraction, the growing rule imprecision becomes a serious problem. At some point the encodings of context and extracted parts become so general that the borders of extractions cannot be identified correctly or even irrelevant sentences are matched by the extraction pattern. The resulting incorrect extractions from the training corpus discredit an extraction rule increasing the uncertainty about its performance on the test corpus. The task of the validation step is to resolve the tradeoff between the rule generality and precision and to decide what induced rules can improve the extraction performance and should be added to the set of correct rules.

The decision whether an induced rule contributes to the extraction quality depends to a large degree on the set of correct rules. If, for instance, there are few general correct rules and a new rule extracts several fragments not covered by the correct rules, it can be validated taking in account some incorrect extractions of this rule. On the other hand, all extractions of an induced rule may be already covered by the correct rules so that its addition will not contribute to the improvement of extraction results.

Let N_i designate the total number and C_i – the number of correctly extracted fragments by the rule r_i. The *rule precision* RP can be defined as

$$RP(r_i) = \frac{C_i}{N_i}$$

A simple and effective method to validate rules involves the usage of a fix precision threshold. Setting it, for example to 0.5 all rules will be validated that

produce at least as many correct as incorrect extractions. However, such inflexible criterion does not account for the complexity of the training corpus, progress of the generalization etc. so that it will often lead to a non-optimal set of correct rules. The validation should therefore be coupled with some objective criteria reflecting the extraction quality.

RP of a rule can be regarded as a *confidence measure* for the validity of its extractions. Since the value of RP represents the relative frequency of correct extractions on the training corpus, it can serve as the *probability measure* for the correctness of extractions made by a rule on the test corpus. While the RP of a rule denotes the probability that a general extraction made by this rule is correct, it is possible to refine this statement for extractions of single attributes (refer to the sec. 9.2.3). Such a confidence measure of the correctness of an extraction can be used, for example, for a human quality assurance closely inspecting the extractions with lower correctness probability.

The overall extraction quality can be measured by recall and precision and the so called *F-measure* (s. next chapter), which combines both other metrics. The purpose of the validation is to improve the extraction quality, which is equivalent to maximizing precision and recall or F-measure. Any validation strategy can therefore be optimized to achieve the maximum values of these evaluation parameters. In the following we present several validation strategies that use F-measure value to optimally evaluate the induced rules.

9.2.2 Rule and Attribute Precision Thresholds

Criteria applied in the evaluation of a whole IE system cannot be transferred in the same manner to the evaluation of a single extraction rule. While the notion of rule precision defined above corresponds with the overall precision measure, the notion of recall cannot be applied to single rules. Since we cannot expect that a single extraction rule covers all extracted fragments in the training corpus, measuring the absolute recall value (the part of fragments correctly extracted by the rule in the total number of expected fragments) is not reasonable. The absolute number of correctly extracted facts and hence indirectly the recall value are controlled by the abstraction degree. Since the increase of abstraction degree and the rule generality is ensured by the induction process, the main criterion for the evaluation of the performance of extraction rules is precision. This can be exemplified considering a set of induced rules with the abstraction degree i: since each of these rules extracts i training examples, the better rules are those that produce less incorrect extractions.

Considering the overall rule precision we neglect the fact that a single rule can extract different attribute values. An overall rule precision value allows only indirect conclusion about the extraction quality of single attributes. It is possible that some rules are especially confident in extraction of certain attributes, e.g. because of good context specification or because of frequently occurring combination and order of other attribute values. Thus it is reasonable to determine and store the precision values for single attributes extracted by a rule. In the application phase when the rule is used to extract information from new texts attribute precisions can serve as the criterion whether the extractions of certain attribute values should be accepted.

Taking both considerations into account, to validate induced rules two thresholds – *rule precision threshold (RPT)* and *attribute precision threshold (APT)* can be used:

$$\forall r_i \in InducedRules\ RP(r_i) \geq RPT \Leftrightarrow r_i \in CorrectRules$$

```
rpt=0.5; apt=0.5; step=0.4;
max_f=f_measure(rpt, apt);
while (step>0.01)
    {prev_rpt=rpt; prev_apt=apt;
     cur_f=explore_direction(step, max_f, rpt);
     if (cur_f==max_f)
        {rpt=prev_rpt;
         cur_f=explore_direction(-step, max_f, rpt);
        }
     max_f=cur_f;
     cur_f=explore_direction(step, max_f, apt);
     if (cur_f==max_f)
        {apt=prev_apt;
         cur_f=explore_direction(-step, max_f, apt);
        }
     max_f=cur_f;
     step/=2;
    }
return (rpt, apt);

explore_direction(step, max_f, threshold)
    {f=max_f;
     while(f==max_f &((threshold+=step))<=1 & threshold>=0)
        {f=f_measure(rpt,apt);
         if (f>max_f) max_f=f;
        }
     if (f<max_f || threshold>1 || threshold<0) threshold-=step;
     return max_f;
    }
```

Figure 9.3: Optimization of RPT and APT for achieving maximum F-measure

The thresholds fulfil different purposes. The RPT assesses how reliable the rule generally is. It establishes a general base for comparison of single rules. If a rule does not surpass the RPT, it will not be validated and will be rejected. The APT qualifies a rule to extract certain attributes allowing for different extracting capabilities of rules with respect to different attributes:

$$AP(r_i, a_j) \geq APT \Leftrightarrow r_i \text{ may extract values of } a_j$$

where $AP(r_i, a_j)$ is the attribute precison of r_i for the attribute a_j.
A rule is qualified for extraction of a certain attribute if the attribute precision of the rule for this attribute is higher than APT. While RPT establishes the actual selection process filtering the rules with low reliability, APT improves the extraction quality examining the extractions of validated rules and confirming only those that are produced by rules that showed the best performance for the respective attribute.

The purpose of the validation is the identification of a subset of induced rules that maximizes the F-measure value (refer to the previous section). RPT and APT are therefore optimized with respect to the F-measure that is achieved by the rules validated with these thresholds. In other words, we are looking for the global maximum of the unknown function $F - measure(RPT, APT)$. The $F - measure$ function selects all induced rules with rule precision bigger than RPT and applies them to the training set. The extractions of a value of an

attribute A are rejected if the attribute precision of the rule for this attribute is less than APT. Remaining extractions are used to calculate precision and recall and, finally, F-measure. The maximum search is complicated by the fact that there are two variable parameters so that the combination of threshold values maximizing the F-measure has to be determined.

We propose an approximating algorithm (s. fig. 9.3) that generalizes the concept of logarithmic search for two parameters. The algorithm can be regarded as a bootstrapping process that consists in finding the local optimum for one parameter (while the value of another is kept constant) and continuing to look for the local optimum for another parameter with the new value of the first one. An important condition is that in every step the value of the F-measure increases. Initializing (RPT, APT) as $(0.5, 0.5)$ the procedure *explore_direction* examines how the F-measure value behaves depending on increase or decrease of the RPT. As long as F-measure value grows, the RPT is modified adding or subtracting a constant difference (which is called *step* in the fig. 9.3). After the local optimum for RPT has been determined, *explore_direction* is used to find the local optimum for APT. In the next iteration of the algorithm the *step* is halved to continue the search in the narrower environment of the found local optima. The algorithm terminates when the *step* value falls below 0.01. This means that the approximation of the maximum value is regarded as satisfactory when the deviation of RPT and APT values from the optimal values RPT_{max} and APT_{max} is less than $1,25\%$.

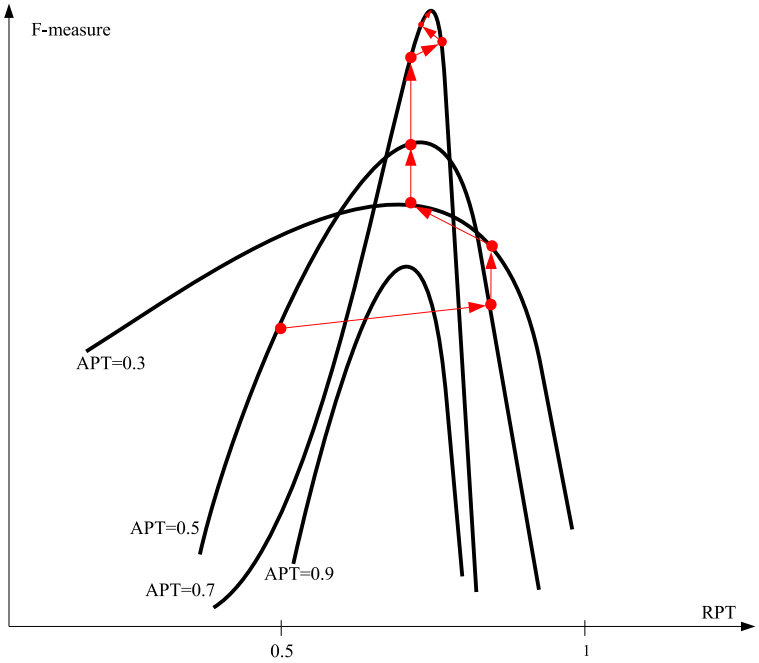

Figure 9.4: Approximation of the maximum F-measure value

The algorithm is not immune to finding local maxima. In an adverse case the *step* may be too small to bridge the gap between a local maximum and a higher F-measure value, which would mislead the algorithm to look only in the narrow environment of the local maximum. However, we did not embed any heuristic treatment of local maxima in the algorithm since the expectation that the approximated F-measure function has one maximum has been confirmed by the empirical results on three examined training corpora. Fig. 9.4 demonstrates typical graphs of the F-measure function. The red arrows illustrate the approximation process. Usually, the maximum is in the vicinity of RPT and APT values that ensure equal precision and recall values.

9.2.3 Local Attribute Precision Thresholds

Continuing the differentiation made in the previous section between the reliability of a rule as a whole and reliability of attribute extraction, one can differentiate between the single attributes. Independent of a concrete rule some attributes can be extracted more reliably than others because of their simple structure, characteristic context etc. (s. sec. 10.2). Instead of evaluating the quality of attribute extraction with one uniform attribute threshold separate precision thresholds for each attribute can be introduced. Using an optimized threshold for every attribute the different extraction quality of single attributes can be considered. As a consequence the recall of attributes with low extraction quality will not suffer from too high and the precision of well extractable attributes – from too low thresholds. The attribute precisions for single attributes extracted by a rule can be regarded, similarly as the RP, as the probability of correct extractions of the values of specific attributes providing even finer confidence measures.

The extraction quality of different attributes within a single rule may significantly differ. An otherwise unreliable rule may be successful in the extraction of a certain attribute. Since local attribute thresholds allow to decide individually for each attribute whether a rule is qualified to extract an attribute value, generally unreliable rules can also be allowed to extract attributes if they surpass the threshold for this attribute. Thus the overall reliability of a rule becomes a redundant measure and can therefore be omitted. Since RPT is no longer considered, only one threshold has to be optimized, which makes the algorithm for maximum approximation simpler. To find an optimal value for a local APT, its values can be linearly traversed in equidistant intervals memorizing the value with the maximum F-measure. If we assume the F-measure function to behave as displayed in fig. 9.4, we can reduce the runtime of the maximum approximation from linear to logarithmic runtime using a modified logarithmic search from 9.3 for one parameter.

Optimized on the training corpus local APTs are used to control the extraction quality during the application phase. For every extracted attribute an extraction rule stores an attribute precision value obtained on the training corpus. If the attribute precision of a rule is higher than the respective local attribute precision threshold, the rule is eligible to extract values of this attribute.

The disadvantage of this evaluation method is that the thresholds values obtained for single attributes may be too customized to the training corpus, producing optimal results on the training texts but being not adequate for other domain texts. Especially attributes with a small number of training instances are prone to a strong bias towards training texts. Since a general attribute precision threshold is determined on a large, representative set of examples, it offers a more reliable validating criterion and ensures a more stable behavior of the extraction system while using local APTs better differentiation and hence optimization of induced rules can be achieved in some cases at the expense of divergent extraction performance in general. Different validating strategies will be therefore subject of evaluation in sec. 12.4.2.

9.2.4 Covering Validation Setup

Techniques relying on optimization of rule and precision thresholds consider only quantitative criteria ignoring the actual rule extractions. A rule that extracts the same fragments that have already been extracted by validated rules will also be regarded as reliable if it achieves satisfactory precision. Covering validation

regards the extractions of an induced rule in conjunction with extractions of other induced rules optimizing the set of extractions made by induced rules.

The induced rules are sorted according to their rule precision. Iterating over the sorted set the best rule is removed and its extractions are evaluated. Only such extractions count as true positive that are correct and have not been made by any induced rule removed earlier. The same effect can be achieved removing the extracted instances from the set of correct extractions. It involves that rules, which extract fragments that are different from other extractions, are stronger rewarded and rules that extract the same fragments as the best rules are punished. Rewarding diversity of induced rules and extraction patterns a better coverage of the test corpus can be expected. Rules with lower precision values that, however, extract distinct fragments that cannot be identified by other rules can also be validated and contribute to a better recall on the test corpus.

During the iteration over sorted induced rule new rule and attribute precision values are calculated based on the number of true positives in the covering setting. RPT and APT are optimized by the algorithm presented in 9.3 with the new rule and attribute precision values. While in the conventional RPT-APT validation the absolute precision values are significant, covering validation optimizes the overall F-measure considering the precision relative to the extractions of other rules and obtaining so more diverse rule set.

The tradeoff in using covering validation is that some rules have a much bigger extracting potential as indicated by the extractions from the training corpus. Since the training corpus provides only limited supply of possible extractions, two rules may produce many common extractions even though they are quite different. The deceptive rule similarity suggested by common extractions may lead to the rejection of reliable rules with a big covering degree and hence to the decreasing of recall and precision values.

9.3 Termination of Rule Induction and Application

9.3.1 Termination

In every iteration step of the induction algorithm increasingly general rules are induced, validated and added to the set of correct rules. When the induced rules become too general to achieve satisfactory precision, the maximum abstraction degree of the extraction rules is reached so that no further induction of reliable extraction rules is possible. This may have two reasons: either the pairs of correct rules for induction of higher abstraction degrees are not similar enough so that no new rules can be induced or the induced rules are not validated failing to pass the precision thresholds.

To ensure that after the i^{th} iteration no rule with a higher extraction degree can be induced or validated, we have to execute the iterations $i + 1 \ldots 2i$ of the algorithm. Since the abstraction degrees of correct rules after the i^{th} iteration lie in the range $[1, i]$, all possible combinations of rules for merging will be covered by the iterations $i + 1 \ldots 2i$. If no new correct rules are induced in these iterations, further generalization by rule merging will not be possible. In this case one last iteration with substitution heuristic is performed trying to improve the recall value.

The induction algorithm terminates when the set of correct rules cannot be altered by any of the generalization heuristics. The set of correct rules and the

optimized attribute precision thresholds are the results of the training stage of GROPUS. The returned extraction rules can be used to extract desired information from any text of the domain of the training corpus.

9.3.2 Application of Learned Extraction Rules

The actual extraction of information is accomplished by matching the extraction patterns with corresponding text passages and executing the extraction action that transfers the attribute value from a text to the target structure. Prior to the matching stage any text is linguistically preprocessed in the same manner as the training texts. After the preprocessing the texts incorporates their content and linguistic information in an XML document (analogously to fig. 5.1). Extraction patterns interpreted as XML queries are executed to retrieve XML fragments containing attribute values. The actual text fragments that are transferred to the target structure are obtained from the textual content of XML fragments.

All rules are generally applied to a text to generate candidate extractions. A rule can only extract an attribute value if its attribute precision determined on the training corpus is greater or equal than the corresponding attribute precision threshold. This mechanism allows to control the extraction quality in the application stage. If a text fragment is extracted by several rules the attribute precision of the best rule is considered. In case that a text contains only one relation tuple (so called "one answer per document" extraction mode) the rule with the highest attribute and rule precisions, matching the text, is selected for extraction of each attribute value respectively. Extractions of all other rules are discarded. In the "one answer per occurrence" mode all rules are eligible to contribute extractions provided they surpass the attribute precision thresholds.