Logical Validation, Answer Merging and Witness Selection
A Study in Multi-Stream Question Answering

Ingo Glöckner & Sven Hartrumpf & Johannes Leveling
Intelligent Information and Communication Systems (IICS)
FernUniversität in Hagen
58084 Hagen, GERMANY
`firstname.lastname@fernuni-hagen.de`

Abstract

The paper presents an approach to multi-stream question answering (QA) using deep semantic parsing and logical validation for filtering answer candidates. A robust entailment check is accomplished by embedding the prover in a relaxation loop. Fallback strategies ensure a graceful degradation of performance in the case of parsing problems. The logical validity score is complemented by false-positive tests and heuristic quality indicators which also affect the selection of the most trusted answers. Separate criteria are used for choosing a suitable 'witness', i.e. a text passage which substantiates the answer. We present two experiments in which the method is applied for merging the results of various state-of-the-art QA systems. The evaluation demonstrates that the approach is applicable to heterogeneous QA streams – in particular it improves results for a combination of precision-oriented and recall-oriented answer streams. The method automatically adapts to these characteristics of a QA system by learning parameters from a training sample.

1 Introduction

Question answering (QA) systems like COGEX (Moldovan et al., 2003), Javelin (Nyberg et al., 2003), InSicht (Hartrumpf, 2005), and QUETAL (Frank et al., 2005) have achieved substantial progress in answering factual questions over large corpora of unrestricted text. Though some of these systems use limited forms of inference, the integration of powerful reasoning capabilities into QA systems is still a challenging task: the systems must balance the number of potential answer candidates that will be explored, and the effort that can be invested into each individual candidate. A solution to this problem is offered by a two-stage design which separates the generation of answer candidates, usually based on shallow linguistic methods, from a subsequent validation stage (possibly) involving more sophisticated techniques like deep semantic analysis and logical entailment testing. This generation/validation paradigm is very popular in QA, and a variety of validation methods have been proposed which include web-based validation (Magnini et al., 2002), fuzzy constraint-based validation (Schockaert et al., 2005), and finally logical answer validation (Raina et al., 2005; Tatu et al., 2006; Glöckner, 2006). The Answer Validation Exercise (AVE) of the CLEF 2006 system evaluation campaign has introduced annotated test collections and an evaluation methodology for answer validation systems. The results of AVE 2006 clearly demonstrate the superiority of logical answer validation compared to alternative methods (Peñas et al., 2006). This suggests that logical validation will also provide the most suitable basis for filtering and merging QA streams in a multi-stream setting.

As pointed out by Magnini et al. (2006), QA@CLEF has seen a consistent increase in the potential benefit to be gained from answer merging. In fact, the relative improvement of an (optimal) combined system compared to the best individual system has grown from 28.4% in 2004 through 34.5% in 2005 to 51.2% in 2006 – probably due to diversification and increasing numbers of participating systems. Thus, there is an apparent promise of multi-stream QA. But there are also methodical arguments in favor of the multi-stream approach: 1) simplified development of additional answer producers due to the availability of an external validation service; 2) better recall (coverage) since heterogeneous systems based on very different techniques for answer finding can be integrated; 3) improved precision because highly specialized, precision-oriented solutions can now be easily combined or added to an existing recall-oriented baseline system. Results are now filtered by the logical validation component, and precision may further benefit from the larger comparison base which permits an aggregation of evidence if an answer was found several times; 4) (potentially) improved processing times because the individual QA systems which contribute answers are independent of each other and can execute in parallel on different machines.

Existing multi-stream QA systems like that of Ahn et al. (2005) are applied to rather homogeneous systems based on shallow processing techniques (like named entity recognition, matching of expected answer types). By contrast, our prime interest is that of combining a precision-oriented system (InSicht) with a recall-oriented baseline system, or specialized solutions for certain question types.

In the paper, we address the central problems which must be solved in order to successfully filter and merge QA streams based on logical answer validation. Specifically, we explain 1) how to increase the robustness of the deep parsing approach by using fallback strategies; 2) how to avoid the construction of a textual hypothesis by directly synthesizing a logical hypothesis representation; 3) how to ensure robustness of the logical entailment test by embedding the prover into a relaxation loop; 4) how to aggregate evidence provided that an answer is supported by several text passages; 5) how to utilize extra-logical quality indicators for breaking ties in answer selection; 6) how to cope with heterogeneous multi-stream setups which include high-precision and high-recall systems; 7) how to adjust the filtering criteria to the QA streams based on an annotated training set; and 8) how to determine the text passage which best supports an answer of interest. Moreover, we consider the question whether separate parameter settings are needed for each QA system or whether it is possible to use a joint model for all QA streams.

2 Merging question-answer streams with MAVE

MAVE[1] (Glöckner, 2006), a system for validating results of QA systems, was developed as a testbed for robust knowledge processing in the MultiNet framework (Helbig, 2006). It was evaluated in the answer verification exercise of QA@CLEF 2006, where it produced the best two runs of the five submitted by all participants for German. Encouraged by these results, we now consider the necessary extensions of MAVE which will turn it into a practical system for merging heterogeneous QA streams.

2.1 System overview

The basic setting comprises a set of *QA streams* generated by given *producers* of answers. An item in such a stream, i.e. a QA result for the question of interest, consists of the answer string

[1]MultiNet-based answer verification

and a supporting *witness* – a text passage which substantiates the answer. These answer-witness pairs can be associated with a confidence score and occurrence count specified by the producer. The answer merging task to be solved by MAVE involves reranking the answer candidates based on the joint evidence of all witness-answer pairs. Robust logical answer validation will be the key method for defining a ranking criterion. Notice that we will assess the quality of reranking by inspecting the *single best answer* determined by MAVE. Apart from answer integration, the merging process also includes the selection of the supporting text passage for each answer in the combined stream, i.e. MAVE also addresses the problem of witness selection.

More specifically, combining QA streams in MAVE involves: a) *Preprocessing*, i.e. application of correction rules to the extracted text passage. These rules will expand collapsed terms like '*Gianni_Versace*' which result from named entity recognition, for example. b) The preprocessed inputs are subjected to a *deep linguistic analysis* by the WOCADI parser (Hartrumpf, 2003) which results in a MultiNet representation (Helbig, 2006). This process includes disambiguation of word meanings, semantic role labeling, a facticity labeling, and coreference resolution. c) The post-processing involves a *normalization* of the generated semantic networks. A list of 1,889 synonym classes for 4,778 lexical concepts is used for replacing lexical concepts with a canonical synset representative.[2] Not only the question and the witness, but also the background knowledge is automatically normalized in this way. d) Next is *query construction*, i.e. the semantic representation of question and answer is translated into a conjunction of query literals which represents the hypothesis expressed by the answer. Notice that different queries will be constructed depending on the parse quality of the involved natural language (NL) expressions (for details see Sect. 2.2.1 and Sect. 2.2.2). e) *Robust entailment proving* is then applied to the constructed list of query literals in order to ascertain that the answer is logically justified by the witness (see Sect. 2.2.2). Inferences are supported by the background knowledge of MAVE: 2,484 basic facts, most of which describe lexico-semantic relationships, and 103 implicative rules which express key properties of the MultiNet relations and knowledge on biographic topics like birth of a person (Glöckner, 2007). f) Additional non-logical *indicators* are extracted, which help detect trivial answers, wrong analyses of the NLP components and other sources of error. g) Every answer-witness pair is assigned an *error level* by combining the results of robust theorem proving and several error indicators. h) Based on an error model derived from a training corpus, the error level is translated into a correctness probability which captures the *justification strength* of an answer given a witness. i) *Justification aggregation* is used to determine a total justification score for an answer based on the combined evidence of all supporting witnesses. j) *Answer selection* is based on this total justification score combined with heuristic indicators of answer quality. k) A *heuristic witness score* is defined which incorporates logical justification strength and additional aspects of witness quality. It is used by MAVE for *witness selection*, i.e. selection of a single text passage which best supports the considered answer.

In the following, these processing steps of MAVE are explained in more detail.

2.2 Logical answer validation

2.2.1 Preliminaries

The most important criterion for filtering and merging of answer candidates in MAVE is the *logical validation* of the answer judging from the given witness (i.e. snippet or combination of snippets). MAVE was originally developed for the answer validation exercise of QA@CLEF 2006,

[2]Apart from its own set of 525 synonym relationships, MAVE incorporates high-quality synonym relationships from HaGenLex (Hartrumpf et al., 2003) and unambiguous cases from OpenThesaurus (`http://www.openthesaurus.de/`).

where answer validation was formulated as a problem of recognizing textual entailments. Thus, the question, e.g. '*Woran starb River Phoenix?*' ("*What caused the death of River Phoenix?*"), and the answer, e.g. '*eine Drogenüberdosis.*' ("*a drug overdose*"), were composed into a hypothesis in affirmative form like '*River Phoenix starb an eine(r) Drogenüberdosis.*' ("*The death of River Phoenix was caused by a drug overdose.*") The task of validating an answer then reduces to proving the hypothesis representation from the logical representation of the suggested witness, e.g. '*'Dark Blood' wurde einfach aufgegeben, als River Phoenix im letzten November vor einem Nachtclub an einer Drogen-Überdosis starb.*' ("*'Dark Blood' was simply given up when River Phoenix died from a drug overdose in front of a night club last November.*")

For the strongly inflecting German language, the construction of hypotheses by simple template filling often results in syntactically defective expressions (in the above example, '*eine*' of the original answer must be changed into '*einer*'). These defects result in bad or failed parses. In order to avoid this problem, we decided to parse the question and the answer separately. The semantic representation of the hypothesis is then *synthesized* from the representations of question and answer, i.e. we will short-cut the construction and parsing of a textual hypothesis and directly work on the logical level.

Example. The MultiNet representation of the question '*Woran starb River Phoenix?*' ("*What caused the death of River Phoenix?*") is translated into a literal list

```
((sub X1 "nachname.1.1") (val X1 "phoenix.0") (sub X2 "vorname.1.1")
(val X2 "river.0") (attr X3 X1) (attr X3 X2) (aff X4 X3) (cstr X4 !FOCUS)
(subs X4 "sterben.1.1"))
```

The analysis of the answer '*eine Drogenüberdosis*' ("*a drug overdose*") is also turned into a literal list, which contains only the literal `(subs !FOCUS "drogenüberdosis.1.1")` in this case. The representation of the logical hypothesis is constructed by renaming variables in the representation of question and witness in such a way that only the `!FOCUS` variable (which refers to the queried entity) is shared by both literal lists, and by subsequently forming the conjunction. In the example, no renaming of variables is necessary, and the literal lists can simply be appended to form the combined query.

Another problem to be solved is the demand for precision of the logical approach which makes logical answer validation potentially very brittle. A single error of the NL analysis module which constructs the semantic representations might result in a failed proof of the hypothesis. To achieve robust logical inference, the MAVE prover is therefore embedded in a *relaxation loop*. The prover keeps track of the longest prefix $L_1, \dots, L_k$ of the ordered list of query literals for which a proof was found. Literals are sorted according to a least-effort criterion (Glöckner, 2006). When a full proof fails, the literal L_{k+1} will be skipped, the remaining literals are reordered, and a new proof of the smaller fragment begins. By subsequently removing 'critical' literals, this process always finds a (possibly empty) query fragment provable from the given knowledge. Consequently it no longer makes sense to describe the result of logical validation by a Boolean decision (true or false). By contrast, the number of skipped literals will serve as an indicator for logical justification strength. The skipped literal count is complemented by indicators which signal other kinds of errors or 'imperfections'. Based on an annotated training set, the system finally switches from computed error counts to the correctness probability of an answer given the witness.

2.2.2 Methods for determining error levels

In order to gain robustness against parsing failures, MAVE offers several methods for determining error levels, the choice of which depends on the quality of the data. If the witness, the question, and the answer all parse well, then the following *s-just* method is used for validation, in which the representations of question and answer are synthesized to a logical query (as sketched in Sect. 2.2.1) which is then passed to the theorem prover. The *s-just* indicators of MAVE comprise:

- `synth-failed-literals`: The number of literals that had to be dropped in the relaxation loop in order to be able to prove the remaining literals from the logical representation of the witness and the background knowledge.

- `synth-missing-constraints`: The number of numerical expressions in the question or in the answer for which no corresponding numerical constraint is present in the constructed logical query. (This situation happens if the parser has failed to build a complete semantic representation of question and answer.)

- `synth-dropped-names` and `synth-name-conflicts` serve to achieve a robust treatment of person names. After analyzing the person names (first names and surnames) found in the constructed query and the witness, MAVE will drop all first names from the query which are not present in the witness, assuming that the person can be identified by its last name. This dropping of a first name in the query in order to make a proof possible is reported by the indicator `synth-dropped-names`. The constructed query and the text might mention persons with the same surname but different first names. In this case, the text passage is likely not relevant to the person of interest. This case is reported by the indicator `synth-name-conflicts` which count the number of such first name conflicts.

- The MultiNet formalism employed for knowledge representation offers a facticity attribute which marks every object and state of affairs as either real, non-real/non-existing or hypothetical/unknown. The indicator `synth-proof-facticity` equals 0 if all entities involved in the proof of the query are real; it equals 1 if at least one involved individual is marked hypothetical; and it equals 2 if at least one involved individual is marked nonreal.

These indicators are combined in the following error count determined by the *s-just* method:

$$\begin{aligned}\texttt{synth-err-count} &= \texttt{synth-failed-literals} + \texttt{synth-missing-constraints}\\ &\quad + \texttt{synth-dropped-names} + \texttt{synth-name-conflicts} + \texttt{synth-proof-facticity}.\end{aligned}$$

An alternative method was introduced to increase robustness against parsing failure. The method will work when the answer cannot be parsed. This case is significant due to simplistic answer extraction methods used by QA systems. The method still requires that the question parses well (which is typically guaranteed since the question is entered by a person and not automatically generated), and also that the witness parses sufficiently well on average (here it will only be demanded that at least one sentence with a full parse or possibly a chunk parse exists in the text passage). The basic idea of the *q-just* method is that of gathering evidence from a proof of the original question. The results of this logical proof are then related to the original answer in the answer-witness pair by a simple matching technique which only requires a lexical analysis but no parse of the answer.

- `question-failed-literals` reports the number of literals which had to be skipped in the relaxation loop in order to find a proof of the remaining literals.

- `question-missing-constraints` counts the number of numerical expressions in the original question which have no counterpart in the constructed logical query. This situation happens when the parser does not find a complete analysis of the question.

- The indicators `question-dropped-names` and `question-name-conflicts` are determined in analogy to the indicators in the *s-just* case.
- `question-proof-facticity` evaluates the proof of the original question that was found by the prover from a facticity perspective (see `synth-proof-facticity`).
- In order to connect the result to the answer suggested by the QA system, MAVE extracts the content words from the answer and computes the corresponding lexical concept IDs (i.e. disambiguated word meanings in HaGenLex) for them. The system then tries to match these concepts with the concept constants found in the 'central sentence' of the witness (see below). Ideally, all concepts can be matched. Thus, the number of mismatches is `answer-total-concepts` − `answer-matched-concepts` in this case, where `answer-total-concepts` is the number of all lexical concepts associated with the answer plus all numeric expressions, while `answer-matched-concepts` is the number of those concepts and numerical expressions that the system was able to match with the central sentence.

The central sentence of the witness is determined as follows. The system collects all discourse entities involved in the proof of the original question (these occur as constants in the instantiated literals). Each such discourse entity gets a vote of 1. If the entity occurs in j sentences of the witness, then it will equally distribute its vote over these sentences, i.e. it will add $1/j$ to the accumulated vote of each of these sentences. The sentence with the highest accumulated vote for all discourse entities involved in the proof of the question will be chosen as the central sentence which forms the basis for the matching of concept constants and numbers as described above. The method deliberately restricts matching to a single sentence in the witness in order to ensure a certain coherence of the matches. Notice that the voting model for determining the most relevant sentence will also be used to calculate an indicator `wn-question-focusing`, which is defined as the accumulated vote of the central sentence divided by the number of voters (i.e. of discourse entities on which the voting was based). This indicator, which measures the agglomeration of the relevant information in the witness, will be used later as one dimension of witness quality (see Sect. 2.4).

Summing up indicator values, we obtain the following error count for the *q-just* validation method, to be used if a parse of the answer fails while question and witness parse well.

$$\begin{aligned}&\texttt{question-err-count} = \texttt{question-failed-literals} + \texttt{question-missing-constraints}\\&\quad + \texttt{question-dropped-names} + \texttt{question-name-conflicts}\\&\quad + \texttt{question-proof-facticity} + \texttt{answer-total-concepts} - \texttt{answer-matched-concepts}.\end{aligned}$$

In the remaining case that the witness and/or the question cannot be parsed, a fixed default justification will be used (see *fallback* case in Sect. 2.2.3).

2.2.3 From error levels to probability of justification

The justification strength of the witness, i.e. the degree to which the witness supports the answer, is determined in the following way. For each validation method $m \in \{\textit{s-just}, \textit{q-just}, \textit{fallback}\}$ and potentially also depending on the QA stream (producer) p, we determine the empirical probability that an answer is correct given that method m is used for evaluation and resulted in error level ℓ. If the *s-just* mode is possible, i.e. question, answer, and witness parse well, then the empirical correctness probability $\mathrm{P}(\textit{answer\_correct}|\texttt{method} = \textit{s-just}, \texttt{err-level} = \ell, \texttt{producer} =$

p) is used for `wn-justification`. If parsing of the answer fails, but the question and witness parse well, then $\mathrm{P}(\mathit{answer\_correct}|\texttt{method} = \textit{q-just}, \texttt{err-level} = \ell, \texttt{producer} = p)$ is used, based on the validation method *q-just* which does not require a parse of the answer. Finally if neither of these methods is applicable since parsing of witness or question failed, then the default correctness probability $\mathrm{P}(\mathit{answer\_correct}|\texttt{producer} = p, \texttt{method} = \mathit{fallback})$ will be used as the fallback justification strength, which is at least sufficient to differentiate between precision-oriented and high-recall systems. These empirical probabilities must be determined from relative frequencies in a *training set* annotated for correctness of answers. In order to better support small training sets, we also considered using the empirical probability $\mathrm{P}(\mathit{answer\_correct}|\texttt{producer} = p)$ (determined from all training items) in the *fallback* case rather than $\mathrm{P}(\mathit{answer\_correct}|\texttt{producer} = p, \texttt{method} = \mathit{fallback})$ (determined only from the smaller set of $\texttt{method} = \mathit{fallback}$ items which cannot be parsed). Results do not change much then, as shown by the evaluation in Sect. 4 and Sect. 5; the corresponding runs are marked by an asterisk (*). Notice that the constraint $\texttt{producer} = p$ will be omitted when a joint error model is used for all QA streams.

2.2.4 Estimation of justification probabilities

Let us now consider the problem estimating these correctness probabilities. Direct calculation of the empirical probabilities $\mathrm{P}(\mathit{answer\_correct}|\texttt{method} = m, \texttt{err-level} = \ell, \texttt{producer} = p)$ from relative frequencies in the annotated training set is not very useful due to a sparse data problem: There might be very few items for a given error level so that using the observed relative frequencies directly would not be reliable. In order to avoid overfitting, we took a different route. The functional dependence of the correctness probability on the error level can roughly be described by an exponential function with negative exponent. We thus fitted an exponential of the general form $\mathrm{f}(x) = \alpha \mathrm{e}^{-\beta x}$ (where $\alpha \in [0,1]$ and $\beta \in \mathbb{R}_0^+$) to the given data. Here we describe the chosen method for fitting the curve which is weighted by the number of supporting items for each error level. The method considers pairs of error levels, determines optimal choices of α and β for each pair, and then selects a weighted median of the choices of α and β determined in this way. More specifically, let $L \subset \mathbb{N}$ be the finite subset of error levels $\ell \in \mathbb{N}$ for which at least one answer-witness pair with $\texttt{synth-err-count} = \ell$ (or $\texttt{question-err-count} = \ell$, depending on the considered method m) exists in the training set. Let n_ℓ denote the number of items in the training set for the considered producer with $\texttt{synth-err-count} = \ell$ (or $\texttt{question-err-count} = \ell$ for the *q-just* method). Further let k_ℓ be the number of all such items which are classified as correct (i.e. supporting a correct answer). We utilize these numbers for a weighted approximation in the following way. Consider all $i, j \in L$ with $i < j$. For the given choice of i, j, we define

$$\beta = \frac{\ln(\max(\frac{k_i}{n_i}, \rho)) - \ln(\max(\frac{k_j}{n_j}, \rho))}{(j-i)} \qquad \alpha = \tfrac{k_i}{n_i} \mathrm{e}^{\beta\, i}$$

where $\rho > 0$ is a small constant which prevents the logarithm from being undefined when the argument equals zero.[3] The resulting α is inserted $n_i + n_j$ times in a bag of all α's and β is inserted $n_i + n_j$ times in a bag of all β's (in practice, two binary trees are used to implement this; the first associates occurrence counts to each α while the other binary tree associates occurrence counts to each β). Having processed all $i, j \in L$ with $i < l$, we fetch the median α' and

[3]In the experiments, $\rho = 0.001$ was used for smoothing. It should be remarked that the choice of ρ will rarely affect results because the median selects a 'middle' value of β while ρ is concerned with the extreme cases.

the median β' from the respective bags (i.e. that middle choice of α' such that half of the entries are smaller than or equal to α' and half of the entries are larger than or equal to α', and analogously for β'). We then use α' and β' to compute $\texttt{synth-just-strengh} = \alpha' \mathrm{e}^{-\beta'\ell}$, where $\ell = \texttt{synth-err-count}$ for the considered item, and analogously $\texttt{question-just-strengh} = \alpha'' \mathrm{e}^{-\beta''\ell}$ for the *q-just* method, where α'' and β'' are determined in a similar way based on $\ell = \texttt{question-err-count}$. Though results can be suboptimal, this method computes the parameters quickly in a single sweep through the data. Moreover, the use of the median promises a certain stability of the method against variation in the data and outliers. Obviously, other methods for computing the parameters (e.g. a least-square fit to k_ℓ / n_ℓ) are also conceivable.

2.2.5 Justification aggregation

So far we only considered the justification strength `wn-justification`, which is fitted to the empirical probability that the answer is correct, given the error level, the method used for computing the level, and the producer of interest. However, `wn-justification` only judges the evidence for the answer which stems from a single witness. In order to obtain a global validity score for the answer from the perspective of every supporting witness, we will now explain the aggregation of this evidence across answer-witness pairs for a given question. Let us define a simplification function σ which maps strings into strings. We define σ such that it drops a list of (non-concept) stopwords; eliminates accents; expands German umlaut characters (e.g. $\sigma(\ddot{a}) = ae$); translates all characters into lower-case; and finally deletes all characters which are not alphanumeric. We use σ for defining an equivalence relation $\sim$ on answer strings where $aw_i \sim aw_2$ iff $\sigma(aw_1) = \sigma(aw_2)$. For a given answer *aw*, we consider all answer-witness pairs (aw_i, wn_i) such that $aw \sim aw_i$ for all $i = 1, \ldots, n$. The aggregated justification strength of answer *aw*, which accumulates the evidence for the answer across compatible answer-witness pairs with respect to $\sim$, is then given by $\texttt{aw-total-justification} = 1 - \prod_i^n (1 - \texttt{wn-justification}_i)$, where $\texttt{wn-justification}_i$ is the justification strength of aw_i judging from witness wn_i. Thus, an answer is unsupported if none of the compatible answer-witness pairs (modulo σ) justifies the answer.

2.3 Heuristic answer scoring and answer selection

Apart from lack of logical justification, there are other reasons why an answer can be wrong. These aspects are captured by false-positive tests. Moreover, additional quality criteria exist which are less important than logical validity or a false-positive criterion but potentially useful for breaking ties when selecting the best answer. Notice that these criteria, unlike `wn-justification`, might differ for answers equivalent under $\sim$. For example, it is possible that one version of the answer can be parsed while another version has a syntactic defect. In this case, answer selection should prefer the well-formed variant.

- An answer should not be incomplete. For the time being, we capture this by a length check and simply set $\texttt{aw-incompleteness} = \min(\texttt{answer-num-chars}/40, 1)$.
- Overlong answers tend to be false. This is captured by a second criterion for answer length, $\texttt{aw-overlength} = \max(0, 1 - \max(\texttt{answer-num-chars} - 40, 0)/80)$.
- Good answers often show high parsing scores, as captured by the `aw-parse-quality` indicator provided by the parser. It ranges from 0 (no parse at all) to 1 (optimal parse).

- Though the justification of the answer might be increased by aggregating many weak justifications, the result must eventually be presented and explained to the user. For that purpose, one needs a single convincing witness, i.e. a supporting text passage which strongly motivates the answer. This concern is captured by `aw-best-wn-quality`, the maximum `wn-quality` of all witnesses supporting the considered answer (see Sect. 2.4 for details).

Moreover, two logical tests for false positives are conducted.

- `aw-not-trivial` asserts that the answer is non-trivial. We call an answer trivial if the logical representation of the answer can be inferred from the logical representation of the question. For example, the logical content of '*Gianni Versace*' (i.e. a person with the name exists) can be inferred from the logical representation of '*Who is Gianni Versace?*'. The criterion will be checked by a precise proof (i.e. without the relaxation loop) because a trivial answer should be 'obviously trivial' and not 'approximately trivial'.

- `aw-not-circular` asserts that the answer is non-circular. An answer in non-affirmative form, especially to a definition question, is circular if the question can be proved from the answer (which should be in non-affirmative form). For example, the question '*What is the Eiffel tower?*' should not be answered by '*The Eiffel tower in Paris*'.

The above criteria are supposed to indicate errors in an answer; this is why they are combined in a conjunctive way (by a multiplication). We use a weighting function $c(w,x) = 1 - w + wx$ for adjusting the relative impact of each criterion on the final quality score. Weights were determined experimentally.

$$\begin{aligned}\texttt{aw-quality-score} &= c(0.1, \texttt{aw-incompleteness}) \cdot c(0.1, \texttt{aw-overlength}) \\ &\cdot c(0.2, \texttt{aw-parse-quality}) \cdot c(0.6, \texttt{aw-not-trivial}) \cdot c(0.6, \texttt{aw-not-circular}) \\ &\cdot \texttt{aw-best-wn-quality} \cdot \texttt{aw-total-justification}.\end{aligned}$$

This quality score is useful for ranking answers of one or more QA streams. However, in the experiments to follow, the quality score will not be used for ranking results but rather for selecting a *single best answer* from the answer candidates. A threshold can be used to cut off answers which fall below a given quality level. This might be useful for sensing that a question cannot be answered given the data. In the later experiments, though, we will focus on the case that a positive answer exists.

2.4 Heuristic witness scoring and witness selection

Merging QA streams (of the assumed kind) not only involves a scoring and selection of answers. If several answer-witness pairs are available for a given answer, then merging also involves the selection of a best witness (relevant text passage) which optimally supports the considered answer candidate. Logical support (as expressed by `wn-justification`) is most important here, but there are further criteria of interest. Some of these criteria, like parsing quality, capture inherent characteristics of the witness passage. Other cases, like the producer's own confidence in a suggested answer, are concerned with the relationship between question, answer, and witness.

- The producers of the answer streams may assign a confidence between 0 (worst) and 1 (best) to each answer candidate in the stream (similar to the QA@CLEF task). The score is assigned to the indicator `producer-score`.

- Producers can assign a number of occurrences to each item; it is assumed that repeated occurrences in the same document will not be inserted as separate items in the answer stream but rather grouped together in a single item. We define a criterion which rewards a high number of occurrences, $\texttt{wn-occurrences} = 1 - 2/(\texttt{occurrences} + 1)$, where `occurrences` is the number of occurrences reported by the answer producer.

- The average parse quality of the witness text should be high. An average is used because the parser assigns parse qualities on a per sentence basis, and the witness is allowed to span several sentences. Thus, `wn-parse-quality` is the average of the parse qualities of the sentences, which ranges between 0 (no parse) and 1 (perfect parse).

- An overlong text passage makes a poor witness. Not only is a longer passage less concise. There is also a higher likelihood that the information checked by the QA system is spread over the text but not coherent. We capture this by the criterion $\texttt{wn-num-sentences} = \max(0, 1 - (\texttt{witness-num-sentences} - 1)/5)$, where `witness-num-sentences` is the number of sentences in the witness.

- The length of the witness is also evaluated in terms of the number of characters. We set $\texttt{wn-num-chars} = \max(0, 1 - \texttt{witness-num-chars}/800)$, where `witness-num-chars` is the character count of the witness.

- The ratio of special characters in the witness should be low, since texts with a very high number of special characters tend to be presented in a schematic way which is hard to parse and interpret automatically. In order to avoid this source of error, we use the criterion $\texttt{wn-special-chars} = \max(0, 1 - 6 \cdot \texttt{witness-special-chars-ratio})$, where `witness-special-chars-ratio` is the number of all characters in the witness divided by the number of special characters (i.e. non-alphabetic, non-digit, non-whitespace printable characters).

- The indicator $\texttt{wn-relativizing-words} = 1/(1 + \texttt{witness-relativizing-words})$ is used to punish occurrences of relativizing words like non-increasing quantifiers ('*few*') or modal expressions, which are hard to interpret on the logical level and a possible source of incorrect results. Here, `witness-relativizing-words` is the relativizing word count of the witness divided by the number of sentences in the witness text. A low value of `wn-relativizing-words` indicates a severe decrease in reliability.

- The MAVE system is equipped with a coreference resolution module (e.g. for interpreting anaphoric pronouns). However, coreference resolution is still prone to errors and a witness which allows the question to be answered from a single sentence is likely more reliable than a passage in which the relevant information is scattered over several sentences. The criterion `wn-question-focusing` already explained in Sect. 2.2.2 measures this agglomeration of the relevant information in a single sentence.

Since the effect of the criteria should be moderated, we used a weighted combination with the above weighting function $c(w,x) = 1 - w + wx$ for defining witness quality:

$$\begin{aligned}\texttt{wn-quality} = {} & c(0.2, \texttt{wn-occurrences}) \cdot c(0.2, \texttt{wn-parse-quality}) \cdot c(0.2, \texttt{producer-score}) \\ & \cdot c(0.1, \texttt{wn-num-sentences}) \cdot c(0.1, \texttt{wn-num-chars}) \cdot c(0.3, \texttt{wn-special-chars}) \\ & \cdot c(0.2, \texttt{wn-relativizing-words}) \cdot c(0.2, \texttt{wn-question-focusing}) \cdot \texttt{wn-justification}.\end{aligned}$$

The `wn-quality` score is applied by MAVE to select a single best witness for each answer of interest. The relevant text passage can be presented to the user as an explanation of the

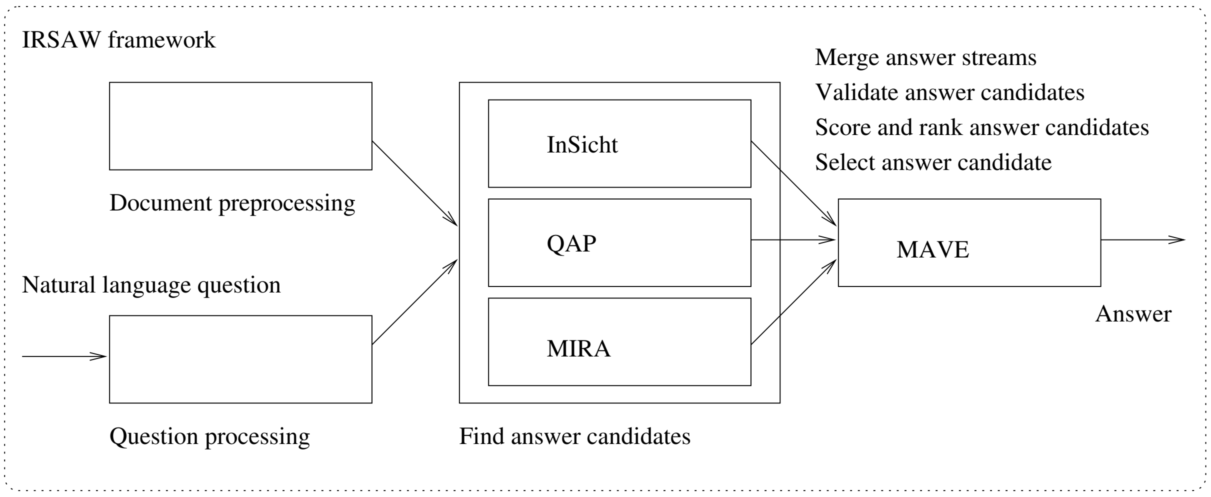

Figure 1: Schema of the IRSAW system.

answer which helps the user understand if the answer is correct and sometimes also clarifies the meaning of the answer.

3 Application in the IRSAW question answering framework

3.1 Description of the IRSAW framework

IRSAW (Intelligent Information Retrieval on the Basis of a Semantically Annotated Web) is a research project in which a modular framework (with the same name) integrating different QA approaches is being developed. Figure 1 shows the general system architecture.

We employed three different methods to create streams of answer candidates: The first answer stream is produced by InSicht, a precision-oriented QA system using a semantic network representation of questions and documents (see Sect. 3.2). The second answer stream originates from a recall-oriented QA system using pattern matching (QAP, see Sect. 3.3), and the third stream results from a modified IR approach (MIRA, see Sect. 3.4). The latter two approaches rely on a preprocessed version of the document corpus (described below), to which sentence boundary detection has been applied on each document to form text segments. These text segments are transformed into XML and indexed in a database management system supporting the tf-idf IR model. Together with the answer validation module MAVE, these modules are part of the IRSAW QA system.

Evaluation is based on the QA@CLEF corpus, which (after duplicate removal) contains 277,000 newspaper and newswire articles from *Der Spiegel*, *Frankfurter Rundschau*, and *SDA (Schweizer Depeschenagentur, Swiss news agency)* from the years 1994 and 1995. The corpus of questions consists of the official German questions employed for the QA task at CLEF in 2004, 2005, and 2006. For each year, a set of 200 questions was released.

3.2 The deep IRSAW producer: InSicht

The deep QA system used for the answer merging experiments is InSicht (Hartrumpf, 2005, 2006), which employs complete sentence parsing, inferences, and semantic representation matching. It comprises six main steps.

In the *document processing* step, all documents from a given collection are transformed into a standard XML format (CES, corpus encoding standard[4]) with word, sentence, and paragraph boundaries marked up by XML elements. All preprocessed documents are parsed by the WOCADI parser (Hartrumpf, 2003) yielding a syntactic dependency structure and a semantic network representation of the MultiNet formalism for each sentence.

In the second step (*query processing*), WOCADI parses the input sentence. Determining the sentence type (here, often a subtype of *question*) is especially important because it controls some parts of two later steps: query expansion and answer generation.

Next comes *query expansion*: Equivalent and similar semantic networks are derived by means of lexico-semantic relations from the computer lexicon HaGenLex (Hartrumpf et al., 2003) and a lexical database (GermaNet), equivalence rules, and inference rules like axioms for MultiNet relations and entailments for situations. The results are disjunctively connected semantic networks that try to cover the representations of many different sentences that (explicitly or implicitly) contain an answer to the question.

In the fourth step (*semantic network matching*), all document sentences matching a semantic network from query expansion are collected. A two-level approach is chosen for efficiency. First, an index of concepts (disambiguated word meanings from HaGenLex) is consulted with the concepts from the question networks. Second, the semantic networks of the retrieved documents are compared sentence network by sentence network to find a match with a question network.

Answer generation is next: NL generation rules are applied to semantic networks matching the query and an NL answer is generated. Therefore answers from InSicht can differ from the exact form found in the documents. The rules also act as a filter for uninformative or bad answers. The results are tuples of generated answer string, one ore more supporting snippets (sentences forming a witness), answer score (a number between 0 and 1), supporting document ID, and supporting sentence ID.

To deal with candidate answers from answer generation (which can be numerous), an *answer selection* step is required at the end. It employs answer clustering and is driven by a preference for more frequent answers and a preference for more elaborate answers. The best answer(s) and the supporting sentences (and the IDs of supporting sentences and documents) are presented.

Due to the relatively strict matching approach, InSicht is precision-oriented: in QA@CLEF 2004, 2005, and 2006 almost no wrong positive answers occurred. But on the other hand, InSicht misses many answers because the parses of relevant sentences contain small errors or a central inference step cannot be made due to lack of background knowledge

The average runtime per question was reduced to around 20 seconds by setting InSicht's time-out per question to 180 seconds. In addition, the maximum of documents investigated during semantic network matching was reduced to 10,000.

The unmodified InSicht system was taken as one answer stream for combining with the recall-oriented shallow approaches described in Sect. 3.3 and Sect. 3.4. In addition, InSicht was modified by introducing and adjusting parameters to provide an alternative answer stream. The corresponding InSicht run is labeled *InSicht (relaxed)* in Sect. 4. The main modifications in the relaxed run are the following:

1. Ignoring semantic layer features. The layer features of the MultiNet formalism describe aspects like the quantification, cardinality, facticity, and genericity of concepts. In the relaxed run, InSicht ignored layer feature differences between question networks and

[4]`http://www.cs.vassar.edu/CES/`

Table 1: Examples for relational triples. Question patterns are translated from German.

Relation name	**Question pattern**
born_in(<key>, <answer>)	Where was <key> born?
died_at_age(<key>, <answer>)	How old was <key> when he/she died?
died_in(<key>, <answer>)	Where did <key> die? / At what age did <key> die?
located_in(<key>, <answer>)	Where is <key>?

document networks. This allows to find additional correct answers, for example when the difference in cardinality does not matter: '*Wahl*' vs. '*Wahlen*' (singular vs. plural of the noun '*Wahl*', "*election*"). On the downside, this can lead to wrong answers, e.g. CLEF question qa05_059 asks about the number of victims of the massacres in Rwanda, while many documents talk about just a single massacre.

2. No answer clustering. InSicht clusters similar answer candidates into one answer candidate representation. This step can make errors or move a better answer candidate into a cluster with a worse answer candidate as the chosen cluster representative. Therefore the relaxed InSicht run employed no answer clustering.

3.3 QA by pattern matching – QAP

QAP (Question Answering by Pattern matching) employs pattern matching on a per-sentence basis, in which answers are found as instantiations of the answer variable. For this approach, about 30 classes of questions were defined. These classes were derived for factoid questions in the CLEF question corpus, forming a relational triple. A relational triple has the form *<relname>(<key>, <answer>)*. *<relname>* is one of *born_in*, *died_at_age*, etc. *<key>* is a set of keywords extracted from the question, and *<answer>* is the answer string to a question corresponding to the relation *<relname>*. Table 1 shows some typical examples for relational triples. Both QAP and MIRA (described in Sect. 3.4) operate with a time-out of 180 seconds per question.

QAP returns the answer exactly as it occurs in the document text. Several large resources were utilized to create question-answer pairs for training the answer identification method. Most important are VERA (Virtual Entity of Relevant Acronyms), an acronym database, and the PND data as used in the German Wikipedia (PND – Personennamendatei, see Hengel and Pfeifer (2005)). An entry in the PND data contains information about a famous person such as his/her place of birth, date of birth, place of death, date of death, aliases, and profession. This data can be transformed to represent question-answer pairs. In addition to explicit information, additional question-answer pairs such as the age at death or meronymy information for locations (typically comma separated values, e.g. '*Seattle, Washington*') can be derived. These question-answer pairs are employed for extracting surface patterns from the CLEF newspaper corpus. For each pair, an IR query containing both question keywords and answer is sent to the sentence database and the first 1000 document sentences are retrieved. (For QAP, no morphologic variants were used.)

Document sentences are tokenized and the token sequences corresponding to question keywords and answer are substituted with keyword and answer variables. For each sentence, the context of the answer variable consisting of two tokens or variables to its left and its right is extracted. Variables are added to the left and right of this context pattern to form a pattern of tokens and variables representing a full tokenized sentence. These patterns are sorted by frequency and the top 1000 patterns are kept for matching. QA in QAP consists of extracting the main keywords for an NL question as well as determining the question type (*relname*). The patterns corresponding to the relation identified are applied to the set of document sentences retrieved. The pattern matching approach returns instantiations of the answer variables, which form strings for answer candidates. Answer candidates contain frequency, score, snippet, answer string, document ID and sentence ID. Scoring consists of a weighted sum of the success rate of pattern matching (precision) with the database score. The top 50 candidates are retrieved for validation by MAVE.[5]

3.4 A modified IR approach for QA – MIRA

MIRA (Modified Information Retrieval Approach for QA) is a recall-oriented approach to QA based on information retrieval combined with the selection of the most frequent word sequence of the expected answer type. Question processing in MIRA consists of the following steps: The NL question is tokenized and stopwords are eliminated to identify the keywords. A naïve Bayesian classifier is applied on features of the question such as the first N word forms. The classifier returns a ranking of expected answer types, of which the top ranked type is selected. The classes for expected answer type are: LOC (location), ORG (organization), PER (person), TIM (time), MEA (measurement and quantity), and OTH (other). All keywords in the NL question are looked up in the CELEX morphology database (Baayen et al., 1995) to find all possible morphologic variants. An IR query is created utilizing all morphologic variants of keywords and submitted to the database system. The top N documents (sentences) are retrieved for further processing ($N = 1000$). The document sentences are then tokenized and their tokens categorized into classes, following the classification scheme mentioned above, i.e. named entities are tagged with LOC, ORG, or PER; temporal expressions (dates) are annotated with TIM, and numeric expressions followed by a unit are associated with MEA. Answer candidates are then preselected by choosing the most frequent word sequences tagged with the expected answer type.

For example, the question '*Who was the first man on the moon?*' is transformed into the IR query '*first man men moon moons*'. Its expected answer type is PER (person). Among the answer candidates for this question are '*Am 21. Juli um 3.56 Uhr mitteleuropäischer Zeit tritt Armstrong als erster Mensch in den Staub des Mondes*' ("*On July 21st at 3.56 o'clock CET Armstrong steps as the first man into the dust of the moon*") and '*Vor 25 Jahren betrat Neil Armstrong als erster Mensch den Mond, doch heute stagniert die bemannte Raumfahrt*' ("*Twenty-five years ago Neil Armstrong was the first man to step onto the moon, but today manned space flight stagnates*"). Both sentences contain a correct candidate answer tagged as a person name ('*Armstrong*' and '*Neil Armstrong*', respectively).

4 Experimental results of answer merging for IRSAW

Let us now describe the experimental results obtained by bundling the three QA streams and merging answers by the MAVE system. As mentioned in Sect. 3.1, evaluation was based on the 600 QA@CLEF questions for the years 2004 through 2006, and the goal of the system was selecting a *single best answer* for each of the questions. Notice that the InSicht, QAP and MIRA systems offer their own means for aggregating scores of answers for which more

[5]Most numeric restrictions have been chosen after empiric observations in testing as well as considering pragmatic factors such as memory requirements and processing time for a more efficient QA system.

Run	#y	#x
InSicht (relaxed)	199.4	10.9
InSicht (relaxed*)	199.2	11
InSicht (optimal)	213	6
InSicht (random)	172.1	14.5
InSicht (standalone)	204	10

Table 2: Results of answer selection for InSicht based on 600 QA@CLEF questions.

than one occurrence in (or justification from) the text is found. However these mechanisms were switched off for the answer merging test. For practical reasons, each system was allowed to suggest no more than 50 answer-witness pairs per question. The three IRSAW producers InSicht (relaxed settings), QAP, and MIRA generated a total of 18,720 answer-witness pairs which covered 571 of the 600 examined questions. All of these answer-witness pairs were annotated for the experiment and 3,553 of these answers (19.0%) were judged correct.

The heterogeneity of the QA streams is witnessed by the following annotation statistics. The InSicht (relaxed) producer contributed 1,212 answer-witness pairs which cover 226 of the 600 questions. The low number of questions for which answer candidates were found shows that deep linguistic processing and semantic network matching in InSicht are relatively brittle, which suggests a combination with more recall-oriented approaches. 625 of the answer-witness pairs produced by InSicht contained a valid answer candidate, which means a precision (before answer selection) of 51.6%. The QAP stream is also rather specialized with respect to the questions that can be answered. QAP generated 2,562 answer-witness pairs which cover only 114 of the 600 questions, however. 1,190 of the answer-witness pairs were judged correct, which means a precision of 46.4%. The other extreme of a recall-oriented system is represented by MIRA which contributed a total of 14,946 answer-witness pairs. MIRA produced 1,738 correct answer-witness pairs which means a precision of 11.6%. Since MIRA produces answer candidates for 520 of the 600 questions, it qualifies as a high-recall method.

In the following, we describe the contents of the individual QA streams with more detail. Since every producer was allowed to submit several answer candidates for each question, we also consider the suitability of MAVE for merging the result of each system in isolation. Notice that in the following, **#y** denotes the number of correct positive answers and **#x** the number of inexact answers. For evaluating answer selection quality of MAVE, we are mainly interested in *relative recall*, i.e. **#y** achieved by MAVE divided by **#y** obtained for optimal answer selection. Therefore the remaining cases where the system cannot select a correct answer are not listed since they are irrelevant for determining relative recall. We further computed a *random baseline* (labeled ‘random’ in the tables), which corresponds to the expected number of correct answers provided a random choice of the preferred answer from the answer candidates available for each question. Generally speaking, the difficulty of answer selection (but also its possible benefit) depends on the difference between the random baseline and the results for optimal answer selection.

The results for the InSicht stream are shown in Table 2. InSicht was run in ‘relaxed’ mode to obtain better recall. In this case, optimal selection of answer candidates will correctly answer 213 of the 600 questions, which marks the upper bound on the number of correct selections that can be achieved. For InSicht, the random baseline achieves as much as 80.8% of relative recall compared to the optimal selection, which once again illustrates the precision-orientedness of the system. All listed MAVE results were averaged over ten runs of ten-fold cross-validation

Run	#y	#x
QAP	48	8
QAP*	47.6	8.2
QAP (optimal)	67	12
QAP (random)	36.2	9.7

Run	#y	#x
MIRA	87	11.8
MIRA*	87.3	11.7
MIRA (optimal)	137	17
MIRA (random)	38.8	6

Table 3: Results of answer selection for QAP (left) and the MIRA stream (right).

(this explains the floating-point numbers in the tables). Here and in the following, the results for two methods of estimating fallback error probabilities are listed. The result labeled 'InSicht (relaxed)' corresponds to the estimation of the default error probability from all items in the training partition with `method` = *fallback*. i.e. either question or witness cannot be parsed. By contrast, the result 'InSicht (relaxed)*' marked with the asterisk was obtained when the fallback error probability was calculated from all items regardless of the assigned `method` (see Sect. 2.4). Compared to the optimal selection, MAVE achieves a relative recall of 93.6% when estimating fallback error probability from items in the training set which cannot be parsed. This means a relative improvement of 15.9% compared to the baseline. When estimating the fallback error probability from the total annotated training set, we get a similar 93.5% of relative recall. The results of the standalone InSicht system using its builtin answer selection are also shown. The 204 correct answers (as compared to 199 found by MAVE) demonstrate that the general-purpose selection method of MAVE comes close in quality to a custom solution for a specific QA system.

The results for the QAP stream are shown in Table 3 (left-hand side). Here, MAVE achieves 71.6% of relative recall in the QAP run, compared to an optimal selection result of 67 questions that can be answered. The result for global fallback errors probabilities in the QAP* run is again very similar (71.0% of relative recall). The random baseline has only 54.0% relative recall. This means a relative improvement of 32.6% for the QAP run compared to the baseline.

The right-hand side of Table 3 lists the results of MAVE for the MIRA stream. Estimation of fallback error probabilities based on the total training set (in MIRA*) achieved slightly better results in this case compared to a computation from only those items which cannot be parsed (MIRA). The best result with **#y**=87.3 correctly answered questions reaches 63.7% of relative recall, which means a relative improvement of 125% compared to the baseline. The benefit of answer validation is especially high in this case since MIRA is a recall-oriented system which produces a large number of answer candidates but only with low precision.

Having discussed the coverage and quality of the individual streams, we now turn to multi-stream combinations. The results of combining InSicht (relaxed mode) and QAP are shown on the left-hand side of Table 4. Compared to the results obtained for the InSicht stream, we get a stable but small improvement of 9 more questions that can be answered. Since InSicht contributes the majority of answers and since QAP is relatively precision-oriented as well, the results 'I+Q (joint)' and 'I+Q (joint*)' using a joint error model for both streams are not very different from the results using separate error models for each stream (i.e. 'I+Q' and 'I+Q*'). As witnessed by the theoretical optimum of 229 questions with at least one correct answer (compared to 213 for InSicht alone), the possible benefit of merging the InSicht and QAP streams is small even for an optimal combination. This suggests that a high-recall stream should be added.

The MIRA stream was designed to fulfill this requirement – it covers a large number of questions due to the use of simple IR-style search and subsequent matching of expected answer types. The results for the combined run of InSicht (relaxed), QAP, and MIRA are shown in

Run	#y	#x
I+Q	208.6	12.1
I+Q*	208.3	12.2
I+Q (joint)	206.9	11.1
I+Q (joint*)	206.5	11.4
I+Q (optimal)	229	8
I+Q (random)	157.1	16.8

Run	#y	#x
I+M+Q	247.4	15.8
I+M+Q*	247.3	15.7
I+M+Q (joint)	209.1	15.6
I+M+Q (joint*)	210.4	15.2
I+M+Q (optimal)	305	17
I+M+Q (random)	94.1	14.4

Table 4: Results of answer selection for the InSicht+QAP streams (left) and for all three IRSAW streams (right) based on the 600 test questions.

Table 4 on the right. The I+M+Q run which combines all three systems with a separate error model for each system achieved the best result in this test, which corresponds to 81.1% of relative recall compared to optimal selection. The runs which use a joint model for all QA streams are significantly worse than the runs with parameters fitted to each individual QA system. For example, I+M+Q (joint) only determined a correct preferred answer for 209 questions, while I+M+Q (with parameters individually adjusted for each QA stream) answered 247 questions correctly. Since MIRA has contributed a large number of answer candidates, it tends to dominate parameter estimation of the joint model. But most correct solutions were produced by InSicht which requires a different choice of parameters suitable for a precision-oriented system.

5 Application to the QA@CLEF AVE 2006 test set

The AVE 2006 test set for German was developed for the evaluation of answer validation systems in the QA@CLEF answer verification exercise (Peñas et al., 2006). It consists of 1,443 answer validation items specified by the following data: a) the original question; b) the affirmative hypothesis; and c) the snippet supporting the answer (i.e. 'witness' in our terminology). The templates used for hypothesis construction by slot-filling are publicly available.[6] In order to fit the AVE test set into the framework assumed here (which avoids the construction of a textual hypothesis in order to improve results for the inflecting language of German), we used the hypothesis templates for a reverse operation of reconstructing the original answer strings from the hypotheses in the AVE test set. The benefits of this approach are apparent from the statistics of parsing qualities: First of all, 93.3% of the questions in the AVE test set can be parsed with optimal quality, compared to only 64.7% of the hypotheses which have a perfect parse. Moreover parsing fails completely for 9.4% of the hypotheses, but only for 1.1% of the questions. This demonstrates the drawbacks of hypothesis construction by template filling for a strongly inflecting language. By contrast, the *q-just* method described above only assumes that a parse of the question exists, which is the case for 98.9% of the questions. The *s-just* method, which synthesizes question and answer representation, is more precise than *q-just.* It requires an additional parse of the answer to exist. Since the answers are syntactically simpler than hypotheses, and typically extracted from a well-formed text rather than constructed by slot filling, parsing quality of the answers tends to be better than that of hypotheses. Thus, we obtain a perfect parse for 83.4% of all non-numeric answers, while parsing fails completely for 7.8% of non-numeric answers. These results are better than those for hypotheses. However, the parser is currently unable to decide on a semantic representation for pure numbers (for example, 1984

[6] `http://nlp.uned.es/QA/AVE/`

Run	#y	#x
First	73.3	3
First*	73.6	2.9
First (optimal)	93	4
First (random)	58.5	3

Run	#y	#x
AVE	89.5	3
AVE*	90	2.6
AVE (joint)	87.8	1
AVE (joint*)	88.3	1
AVE (optimal)	122	1
AVE (random)	49.3	1.7

Table 5: Results of MAVE for the best individual system (left) and for all QA streams of AVE 2006 (right) based on the 200 questions of QA@CLEF 2006 for German.

might refer to a year or simply to a positive integer). Therefore MAVE falls back on the *q-just* method for numbers, which comprise 9.8% of all answer strings. We intend to add a special treatment of this important case.

The AVE test set for German consists of the eight QA streams produced by the QA@CLEF participants for the language pairs DE-DE (three systems with two runs each) and EN-DE (one system with two runs), see (Magnini et al., 2006). The official annotation of the AVE test set classifies each validation item as either YES (344 correct answers supported by the witness), NO (1,064 invalid answers judging from witness), or UNKNOWN (35 borderline cases). In order to fit this annotation into our framework, an answer classified YES for at least one validation item was considered correct ('**y**'); an answer classified NO in all validation items was judged wrong ('**n**'); the remaining cases were inspected manually and usually classified inexact ('**x**').

Let us again start by describing the results of individual systems. We shall focus on the results of the best individual QA system which contributed results to the test set, shown in Table 5 (left). Notice that systems were allowed to specify more than one answer candidate per question, and we are again interested in selecting a single best answer for each question. The table lists the baseline (expected result for random choice from candidate answers), the optimal result (for a virtual selection system which always chooses a correct answer if available), and the results of MAVE (estimating default error probabilities from the *fallback* method cases in the training partition, or from all items in the First* case, again averaging over ten runs of ten-fold cross-validation). For the best individual system, results of MAVE are half-way between the baseline and optimal answer selection.

Finally, the right-hand side of Table 5 shows the results when MAVE is applied for selecting a best answer from all available candidates in the AVE test set. In this case, an optimal combined system might answer 122 questions by selecting from the answer candidates. Compared to the results for the best individual system shown in Table 5 (left), there is a lot to be gained by merging QA streams in this case. However, random choice from the candidate answers for a given question would correctly answer only 40.4% of questions on average (see baseline result). This is due to the fact that the AVE test set contains only 23.8% of correct answer-witness pairs, i.e. the QA producers tend to be recall-oriented. MAVE successfully filters and merges these QA streams. The results (obtained by 10-fold cross-validation) show a relative improvement compared to the baseline of up to 82.6%. Moreover, results are better than those of any individual participating system. The best results (90 questions correctly answered, i.e. 73.8% of relative recall compared to the optimal selection) were obtained when determining fallback error probabilities from all training items and when using a separate error model for each QA system. Interestingly, using a joint error model for all QA streams had only a minor effect on the quality of results in this case. We attribute this to the fact that the AVE test set is dominated by a few QA systems with similar characteristics. The best systems were also those which contributed most answer candidates, so that fitting parameters to the training set as a whole also resulted in a good choice of parameters for the relevant QA systems. Notice that by manually adjusting the α and β parameters introduced in Sect. 2.2.4, we were able to correctly select answers for 93 questions (76.2% of relative recall compared to optimal selection). This demonstrates that the method for parameter fitting works pretty well since the results for estimating parameters in cross-validation come close to the results of manual optimization.

6 Conclusion

We have described the MAVE architecture for answer merging and witness selection which is particularly suited for combining heterogeneous QA sources. The system adapts to the quality of the inputs. Thus, if the input is grammatical, then deep methods and deep indicators will be used. The system automatically switches to robust pattern-matching when an answer cannot be analyzed on the logical level. Based on an annotated training set, the validation method can automatically adapt to high-precision and high-recall QA systems. The proposed method for parameter selection adjusts the parameters in a single sweep through the training set. Though the determined parameters are not necessarily optimal, the method is confirmed by the good performance of MAVE and by the stability of results under cross-validation. The approach was evaluated for two multi-stream QA setups. The IRSAW experiment included the precision-oriented InSicht stream, the specialized QAP stream for limited question classes (e.g. questions for acronyms), and the recall-oriented MIRA stream. Due to the lack of balance in the test set (most answer candidates were produced by MIRA while most correct answers originated from InSicht), the use of a joint model resulted in poor performance. However, we obtained good results for the combined IRSAW run when using custom parameters for each producer. The second experiment based on the AVE 2006 test set for German was also successful: A joint model for the eight QA streams was sufficient to achieve an improvement of 82.6% compared to the baseline and outperform any individual QA system represented in the test set. In the future, we will conduct ablation experiments in order to determine the factors in the MAVE architecture which show the strongest impact on performance scores. Moreover, we will replace the heuristic methods of the current design with a more uniform probabilistic model.

Acknowledgments

Research described in this paper was in part funded by the DFG (Deutsche Forschungsgemeinschaft) in the project IRSAW (Intelligent Information Retrieval on the Basis of a Semantically Annotated Web).

References

Ahn, D.; Jijkoun, V.; Müller, K.; de Rijke, M.; Schlobach, S.; and Mishne, G. (2005). Making stone soup: Evaluating a recall-oriented multi-stream question answering system for Dutch. In *Multilingual Information Access for Text, Speech and Images: 5th Workshop of the Cross-Language Evaluation Forum, CLEF 2004*, pp. 423–434. Berlin: Springer.

Baayen, R. H.; Piepenbrock, R.; and Gulikers, L. (1995). *The CELEX Lexical Database. Release 2 (CD-ROM).* Philadelphia, Pennsylvania: Linguistic Data Consortium, University of Pennsylvania.

Frank, A.; Krieger, H.-U.; Xu, F.; Uszkoreit, H.; Crysmann, B.; Jörg, B.; and Schäfer, U. (2005). Querying structured knowledge sources. In *Proc. of AAAI-05, Workshop on Question Answering in Restricted Domains*, pp. 10–19. Pittsburgh, Pennsylvania.

Glöckner, I. (2006). University of Hagen at QA@CLEF 2006: Answer validation exercise. In *Working Notes for the CLEF 2006 Workshop*. Alicante, Spain.

Glöckner, I. (2007). Filtering and fusion of question-answering streams by robust textual inference. In *Proc. of KRAQ'07*. Hyderabad, India.

Hartrumpf, S. (2003). *Hybrid Disambiguation in Natural Language Analysis*. Osnabrück, Germany: Der Andere Verlag.

Hartrumpf, S. (2005). Question answering using sentence parsing and semantic network matching. In *Multilingual Information Access for Text, Speech and Images: Results of the Fifth CLEF Evaluation Campaign*, volume 3491 of *LNCS*, pp. 512-521. Berlin: Springer.

Hartrumpf, S. (2006). Extending knowledge and deepening linguistic processing for the question answering system InSicht. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum, CLEF 2005*, volume 4022 of *LNCS*, pp. 361–369. Berlin: Springer.

Hartrumpf, S.; Helbig, H.; and Osswald, R. (2003). The semantically based computer lexicon HaGenLex. *Traitement automatique des langues*, 44(2):81–105.

Helbig, H. (2006). *Knowledge Representation and the Semantics of Natural Language*. Berlin: Springer.

Hengel, C. and Pfeifer, B. (2005). Kooperation der Personennamendatei (PND) mit Wikipedia. *Dialog mit Bibliotheken*, Jg. 17, H.3:18–24.

Magnini, B.; Giampiccolo, D.; Forner, P.; Ayache, C.; Osenova, P.; Peñas, A.; Jijkoun, V.; Sacaleanu, B.; Rocha, P.; and Sutcliffe, R. (2006). Overview of the CLEF 2006 multilingual question answering track. In *Working Notes for the CLEF 2006 Workshop*. Alicante, Spain.

Magnini, B.; Negri, M.; Prevete, R.; and Tanev, H. (2002). Comparing statistical and content-based techniques for answer validation on the web. In *Proc. of the 8th AI*IA*. Siena, Italy.

Moldovan, D.; Clark, C.; Harabagiu, S.; and Maiorano, S. (2003). COGEX: A logic prover for question answering. In *Proc. of NAACL-HLT 2003*, volume 1, pp. 87–93. Edmonton, Canada.

Nyberg, E.; Mitamura, T.; Callan, J.; Carbonell, J.; Frederking, R.; Collins-Thompson, K.; Hiykumoto, L.; Huang, Y.; Huttenhower, C.; Judy, S.; Ko, J.; Kupść, A.; L. V. L.; Pedro, V.; Svoboda, D.; and Durme, B. V. (2003). The JAVELIN question-answering system at TREC 2003: A multi-strategy approach with dynamic planning. In *Proc. of TREC 12*.

Peñas, A.; Rodrigo, A.; Sama, V.; and Verdejo, F. (2006). Overview of the answer validation exercise 2006. In *Working Notes for the CLEF 2006 Workshop*. Alicante, Spain.

Raina, R.; Haghighi, A.; Cox, C.; Finkel, J.; Michels, J.; Toutanova, K.; MacCartney, B.; Marneffe, M.; Manning, C.; and Ng, A. (2005). Robust textual inference using diverse knowledge sources. In *Proc. of the First PASCAL Challenges Workshop*.

Schockaert, S.; Cock, M.; and Kerre, E. (2005). Fuzzy constraint based answer validation. In *Advances in Web Intelligence (AWIC 2005)*, pp. 394–400. Berlin: Springer.

Tatu, M.; Iles, B.; and Moldovan, D. (2006). Automatic answer validation using COGEX. In *Working Notes for the CLEF 2006 Workshop*. Alicante, Spain.