University of Hagen at QA@CLEF 2006: Answer Validation Exercise

Ingo Glöckner
Intelligent Information and Communication Systems (IICS)
University of Hagen (FernUniversität in Hagen)
58084 Hagen, Germany
ingo.gloeckner@fernuni-hagen.de

Abstract

The paper features MAVE (MultiNet answer verification), a system for validating results of question-answering systems which serves as a testbed for robust knowledge processing based on multi-layered extended semantic networks (MultiNet [4]). The system utilizes logical inference rather than graph matching for recognising textual entailment, which makes it easily extensible by further knowledge encoded in logical axioms. In order to ensure robustness, the prover is embedded in a relaxation loop which subsequently skips 'critical' literals until a proof of the reduced query succeeds. MAVE uses the number of skipped literals as a robust indicator of logical entailment. Although the detection of 'critical' literals does not necessarily result in the minimal number of non-provable literals, the skipped literal indicator performed very well in the AVE experiments. MAVE parses the hypothesis strings and is therefore sensitive to syntactic errors in hypothesis generation. A regular correction grammar is applied to alleviate this problem. The system also parses the original question (when available) and tries to prove it from the snippet representation. The skipped-literal count for the question is then combined with a similarity index for hypothesis vs. found answer. This method boosts recall by up to 13%. Additional indicators are used for recognizing false positives. The filter increases precision by up to 14% without compromising recall of the system.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Linguistic processing*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering, Selection process*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Performance evaluation (efficiency and effectiveness)*; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Inference engines*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Predicate logic, Semantic networks*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

Keywords

Answer Validation, Question Answering, Recognising Textual Entailment (RTE), Robustness, MultiNet

1 Introduction

1.1 System description

MAVE is a knowledge-based system for answer validation which uses deep linguistic processing and logical inference for recognising textual entailment. The system assumes XML input as specified in the AVE 2006 description. A validation candidate *p* consists of a hypothesis string constructed by substituting the answer of the QA system into the original question, and by the extracted snippet which contains the relevant text passage. Optionally, the question string can also be given. MAVE uses a standard cascaded architecture for processing validation candidates. The sequence of processing steps involves preprocessing, linguistic analysis, and indicator extraction (the results of logical inference are also represented by indicators). The final aggregation step determines the `YES`/`NO` decision and confidence score from a combination of the indicators. These processing stages and the involved components of MAVE are detailed in the following.

Preprocessing The hypotheses are very often syntactically ill-formed or semantically defective due to the primitive mechanism used for generating hypotheses from fixed templates. For example, the hypothesis might contain the phrase *'im Jahre 26. April 1986'* (En: *'in the year April 26, 1986'*) rather than listing the date only. Sequences of two prepositions are also often found, e.g. *'in in Lillehammer'* rather than *'in Lillehammer'*. These errors deteriorate the results of syntactic parsing, and a set of regular expressions is therefore applied in the preprocessing stage which corrects many of them. For example, sequences of local prepositions are reduced to a single local preposition. The snippet strings often need correction as well. Syntactic errors in the snippets mostly result from wrong XML-text conversion (contents of XML elements are concatenated without proper punctuation), from wrong segmentation (snippet extraction disregards sentence boundaries), from named entity recognizers which collapse multi-word units into new tokens like *'Gianni_Versace'*, and from primitive mechanisms for reference resolution which simply replace pronouns by a named entity they most likely refer to. Some of these errors can be fixed by regular correction patterns, but the detoriation of extracted snippets by artefacts of the QA systems remains an issue.

Linguistic analysis The WOCADI parser [1] is used for natural-language analysis. Linguistic processing starts with a morphological and lexical analysis, i.e. lemmatization, decomposition of noun compounds, and assignment of possible word meanings. These word meanings are expressed by lexical concept constants from the HaGenLex computational lexicon for German [3]. WOCADI then attempts a deep syntactic analysis, but it is robust against non-grammatical input and often returns a chunk parse (incomplete parse) in the event of such syntactic errors. The parser also constructs a semantic representation of the natural language expression, which is expressed in the MultiNet formalism [4] (a modern dialect of semantic networks). This process includes the disambiguation of word meanings, semantic role labeling, and facticity labeling of all discourse entities (which helps to discern real situations from hypothetical or negated cases). WOCADI handles intrasentential coreference resolution and also prepares the necessary data for a subsequent intersentential coreference resolution (e.g. for interpreting referring pronouns). The output format of the WOCADI parser, which contains the results of all analysis levels, is documented in [2].

Post-processing of representations and query construction Snippets are allowed to span multiple sentences. The representations of individual snippet sentences must hence be integrated into a global meaning representation of the snippet text [5]. This assimilation process involves a resolution of intersentential anaphora, which is based on the coreference information generated by WOCADI. The post-processing further involves a simplification and normalization of the generated semantic networks. A list of synonyms and near-synonyms is used for replacing all lexical concept constants with the canonical synset representative. This eliminates the need of handling synonyms on the level of knowledge processing. The resulting semantic network for a snippet is represented as a logical conjunction of elementary facts which correspond to the edges of the network and to additional node attributes. While the snippet facts are dynamically added to the knowledge base, the hypothesis representation (or the question representation) must be turned into a logical query, i.e. a conjunction of literals in which all entities mentioned in the hypothesis are represented by variables. A proof of such a literal will bind the variables to the appropriate discourse entities mentioned in the snippet or to a skolem term built from such entities. The resulting logical query is further refined by eliminating complicated constructions. This process eliminates modalities *'müssen'* (En: *'must'*) and *'können'* (En: *'can'*), for example.

Indicator extraction A total of 46 indicators is extracted from the hypothesis, the snippet and from the question string, which include character-based indicators (like the length of the snippet or the proportion of non-alphabetic characters), lemma-based indicators (like the `hypo-triviality` ratio described below), concept-based indicators (like the presence of concepts which indicate negative polarity contexts), syntax-based indicators (e.g. number of sentences in the snippet), indicators computed from the semantic network representation (like the indicator `hypo-lexical-focus` described below), and finally proof-based and matching-based indicators. It was not yet possible to consider all of these indicators in the scoring function of MAVE because there was no development collection for German. The eleven indicators which were chosen for the scoring criterion are explained at the end of this section.

Robust entailment proving The most important indicators for deciding answer correctness are determined by the MAVE prover, an improved version of the resolution-based inference engine for MultiNet described in [5]. The theorem prover expects queries to be expressed as a conjunctive list of literals. The knowledge base is comprised by variable-free facts (these describe the MultiNet representations of the snippet, as well as lexico-semantic relationships and bindings of rule schemata), and by implicative rules which derive a conjunction of conclusion literals from a conjunction of premise literals. All variables which only occur in the conclusion are considered existentially quantified. Because all such variables are replaced with skolem terms, application of an axiom will always result in a fully instantiated (variable-free) conclusion.
The prover uses iterative deepening to find answers, i.e. a bounded depth first search with depth increment after each iterative deepening cycle. The iterative deepening process is stopped when a given maximum depth limit of 5 is reached or when a time limit is exceeded (20 seconds in the experiments). Performance of the prover is optimized by term indexing techniques and by literals sorting. Thus, the literals are assigned costs according to the estimated effort for proving the literal given the current instantiation pattern of the literal, e.g. pattern `(sub var non-var)` for the literal `(sub X lesen.1.1)`. The literals are then ordered in increasing order of estimated effort, where the estimate for each literal L_i takes into account the instantiation pattern of bound vs. variable arguments which results from proving literals $L_1, \ldots, L_{i-1}$.
In order to achieve robust logical inference, the theorem prover is embedded in a relaxation loop which drops part of the conjunctive query when a proof of the current query fragment fails or can not been found within the time limit. By subsequently removing 'critical' literals from the query, this process always finds

a (possibly empty) query fragment provable from the assumed knowledge. The number of skipped query literals which are not contained in the provable fragment will be used as a numeric indicator of logical entailment which is robust against slight errors in the logical representation of hypothesis and snippet and also against gaps in knowledge modeling. The selection of the 'critical' literal (which will be skipped in the next iteration of the relaxation loop), is based on the least-effort ordering of the literals. The prover keeps track of the longest prefix $L_1, \ldots, L_k$ of the ordered literal list for which a proof was found. When a full proof fails, the literal L_{k+1} will be skipped and a new proof of the smaller fragment will be started. Notice that literals $L_{k+2}, \ldots, L_N$ must be re-ordered because the instantiation patterns can change when deleting L_{k+1} from the literal list. The relaxation loop does not necessarily determine the smallest possible number of non-provable literals. This will not affect the success of the approach in practice, however (see Sect. 3).

Aggregation of indicators An indicator evaluation language was developed which makes it easy to experiment with scoring criteria. In this language, all indicators can be treated like variables which are automatically bound to the appropriate value for each considered item in the test collection. In addition, aggregation functions like maximum, average etc. are available in the scoring language. The language also offers means for dealing with undefined indicator values. Expressions in this scoring language can then be translated into active scoring functions (which are ordinary functions in the Scheme programming language). Both the `YES`/`NO` decision of the run and the confidence score are computed from expressions in this scoring language.

1.2 Knowledge resources

The background knowledge base of the first MAVE prototype comprises 2,484 basic facts which interrelate 4,666 lexicalized concepts. About 2,000 of these lexico-semantic relationships were supplied by the computational lexicon HaGenLex. A typical example is the MultiNet relation `CHEA`, which holds between events described by verbs like *'aktivieren'* (En: *'to activate'*) and the corresponding abstractum *'Aktivierung'* (En: *'activation'*). The remaining 500 facts capture important relationships not covered by HaGenLex, e.g. about the connection between states (*'Spain'*), state adjectives (*'Spanish'*), and inhabitant names (*'Spaniard'*). Apart from the facts, MAVE uses 103 implicative rules. 38 of these rules are so-called 'R-axioms', which define key properties of the MultiNet relations. The remaining 65 rules include meaning postulates for frequent verbs not covered by the HaGenLex relations, and also a rudimentary domain modelling which covers a few important topics like the birth and death of a person, being awarded a prize, and being a member or a leader of an organization. These topics were identified based on the known questions of QA@CLEF 2004 and 2005.
HaGenLex is also the source of more than 1,000 high-quality synonyms which are used for normalizing concept names. 525 additional synonyms or near-synonyms for frequent terms were manually compiled from OpenThesaurus synsets. The resulting system comprises 636 synonym classes. Not only the representations of query and snippet, but also the fact knowledge base and the implicative axioms are automatically normalized by replacing known synonyms with a canonical representative for the synset. This normalization increases coverage of this model because distinct lexical constants are mapped to a single canonical representation. The background knowledge which refers to the canonical representation is then applicable to all lexical variants of expressing the underlying concept.
Due to lack of time, it was not possible to integrate additional resources like GermaNet. This means that knowledge modeling in the first MAVE prototype is rather sparse. However, most of the hypothesis-snippet

entailments or question-snippet entailments in the AVE test set are not very knowledge intensive, and the robust inference mechanism of MAVE is specifically designed for absorbing a few gaps in knowledge modeling.

1.3 Indicators and scoring function

Defining the scoring criterion The following basic scoring criterion is used by MAVE. The returned number reflects the imperfection score of the considered hypothesis-snippet pair. A perfect logical entailment is scored 0, while an entailment relationship with slight errors (detailed below) will be scored by a small integer depending on the number of detected mismatches. A large value of 1000 will be returned to indicate that the pair should be rejected in any case. The imperfection score `imp-score`(p) of hypothesis pair p is given by the expression:

```
(max
 (* (max (? (>= hypo-triviality 0.8) 0) (= hypo-num-sentences 2)) 1000)
 (min
  (? (max (* (max (< hypo-collapse 0.7) hypo-lexical-focus) 1000)
          (+ hypo-missing-constraints hypo-failed-literals))
   1000)
  (? (max (* question-lexical-focus (= hypo-delta-matched 0) 1000)
          (+ question-failed-literals
             question-missing-constraints
             (- hypo-delta-concepts hypo-delta-matched)))
   1000)))
```

Indicator evaluation was implemented in the Scheme programming language which explains the prefix notation of operators in the above expression. Note that the indicators can be undefined in certain cases. The functions `max`, `+`, `-` and `*` are assumed to be defined only if all arguments are defined. The special expression `(? A B)` is used to handle undefined values. It returns `A` if `A` is defined and `B` otherwise.
The basic indicators on which the computation of the `imp-score` is based are defined as follows.

- `hypo-triviality` – *Lemma repetition ratio.*
  This criterion serves to identify trivial hypotheses like *'Gianni Versace was Gianni Versace'* which are logically valid but not acceptable as an answer. In order to detect such wrong positives, the `hypo-triviality` criterion measures the ratio of lemmata which occur an even number of times.[1] Trivial hypotheses of the above kind result in a high repetition score. Empirically, a threshold of `hypo-triviality` ≥ 0.8 was found to identify such hypotheses reliably. In the case that there is no sufficient data for computing the `hypo-triviality` indicator (because the syntactic processing failed completely), the assumption of a non-trivial hypothesis is a reasonable default. This assumption is captured by the expression `(? (>= hypo-triviality 0.8) 0)`, i.e. no error condition by default.

- `hypo-num-sentences` – *The number of sentences in the hypothesis.*
  Due to odd punctuation and syntax errors, the parser might wrongly consider a hypothesis as consisting of two sentences although in fact we only have single-sentence hypothesis. We therefore know that there is something wrong with the analysis of the parser if `hypo-num-sentences` $= 2$. Such hypotheses get the default evaluation `NO`.

[1]Certain lemmata like German *'sein'* (En: *'be'*) will be filtered out before determining this ratio.

- `hypo-collapse` – *Query extraction control index*
  When there is no perfect parse of a sentence, the result of semantic analysis can be incomplete. Compared to the correct query representation, the constructed logical query will contain too few literals in this case and thus be underspecific. In order to detect this cause of false positives, we define a query extraction control measure
  $$\texttt{hypo-collapse} = \frac{\texttt{\#hypothesis-literals}}{\texttt{\#hypo-content-words}}$$
  which relates the size of the constructed query (number of literals in the representation of the hypothesis) and the number of content words in the hypothesis string. A value of `hypo-collapse` < 0.7 was found to be a reliable indicator for the case that the logical representation is too small compared to the size of the hypothesis. A proof of the logical hypothesis is not useful for judging answer correctness in this case because it likely corresponds to a false positive.

- `hypo-lexical-focus` – *Hint at potential false positive*
  This binary criterion identifies a situation where the logical hypothesis representation is too permissive compared to the intended meaning of the hypothesis. This problem in query construction affects mainly hypotheses involving measurable properties (like *'8585 meters high'*). A modification of the parser will eliminate the need to consider this case in the future.

- `hypo-missing-constraints` – *Control index for numerical constraints*
  Apart from regular words, the hypothesis can also contain numbers. These numbers must occur as restrictions in the constructed logical representations (for example, a restriction of the year to *1994*). Although the syntactic-semantic parser normally treats these numeric constraints as expected, it turned out that they are occasionally dropped when a complete parse of the sentence fails. The indicator `hypo-missing-constraints` was introduced to recognize this case. It is defined as
  $$\begin{aligned}\texttt{hypo-missing-constraints} = {} & \texttt{\#numerical-tokens-in-hypothesis} \\ & -\texttt{\#numerical-constraints-in-logical-hypothesis}.\end{aligned}$$
  The `imp-score` criterion will treat each missing constraint like a skipped literal which could not be established by logical inference.

- `hypo-failed-literals` – *Number of non-provable literals*
  As explained above, the prover is embedded in a relaxation loop which skips non-provable literals until a proof of the rest literals succeeds. In this way, we obtain the indicator `hypo-failed-literals` (number of non-provable literals in the hypothesis representation). A value of 0 means strict logical entailment, while a small integer indicates a few mismatches or knowledge gaps.

- `question-failed-literals` – *Number of non-provable literals for the question*
  The MAVE system also tries to prove the original question (rather than the hypothesis) from the snippet. The indicator `question-failed-literals` counts the number of literals in the logical question representation which had to be skipped in order to prove the remaining question literals.

- `question-missing-constraints` – *Control index for numerical constraints in the question*
  The indicator counts the number of numeric tokens in the original question which are not expressed in the logical representation of the question. The evaluation criterion `imp-score` treats these missing

constraints like a skipped literal which cannot be established in the logical proof. See indicator `hypo-missing-constraints` for motivation.

- `question-lexical-focus` – *Hint at potential false positive*
  Due to a bug in semantics construction, the logical representation of questions involving measurable adjectives is too permissive in a few cases, which means a higher risk of a false positive. These situations are detected by the binary indicator `question-lexical-focus`, the analogue of `hypo-lexical-focus` for questions. The problem will be fixed in future revisions of MAVE and the indicators can then be omitted.

- `hypo-delta-concepts` – *Number of answer words in the hypothesis*
  A proof of the original question will only support the given hypothesis if the answer from which the hypothesis has been built is similar to the answer found in the proof of the question. As the basis for establishing such a similarity, we extract the answer part of the hypothesis (knowing that the hypothesis has been constructed from the question and from the original answer determined by the QA system). `hypo-delta-concepts` counts the number of content-bearing tokens in the hypothesis which likely stem from the original answer of the QA system and not from the question.

- `hypo-delta-matched` – *Number of matched answer words*
  In order to relate the answer of the QA system to a proof of the original question based on the snippet, we use the matching coefficient `hypo-delta-matched`. Based on the proof of the logical question representation over the snippet and the known origin of literals in one of the sentences of the snippet, we first identify that sentence in the snippet which contains most of the literals supporting the proof. The content-bearing and numeric tokens in the answer part of the hypothesis are then matched against the lexical concepts and cardinality specifications in the MultiNet representation of this central sentence.[2] In this way, we obtain the number of those answer words which can be matched with the selected snippet sentence. The focus on a most relevant sentence of the snippet helps detect the case that a hypothesis refers to parts of the snippet not related to the answer.

Defining the selection criterion A simple threshold θ is used to determine YES/NO decisions from the basic evaluation score as defined above, i.e.

$$\mathtt{select}(p) = \begin{cases} \mathtt{YES} & : \mathtt{imp\text{-}score}(p) \leq \theta \\ \mathtt{NO} & : \text{else}\,. \end{cases}$$

Thus all hypothesis-snippet pairs with an imperfection score `imp-score` $\leq \theta$ will be evaluated `YES`, while those items with more than θ imperfections will be rejected.

Defining the confidence ranking The assignment of confidence scores is based on the assumption that for `YES` decisions, the results with the least number of imperfections (as expressed by the evaluation criterion) are most reliable. Specifically, results with an imperfection score of 0 (no mismatches at all) should be fully reliable with score 1, and all other `YES` decisions should get smaller but positive confidence evaluation which depends on the number of detected errors. A simple linear function is used to obtain this behaviour. As to the `NO` decisions, a larger number of imperfections (non-provable literals) means more evidence that the answer is indeed false. Assuming that σ or more non-provable literals (and other imperfections) mean

[2]Matching is performed on the level of lexical concepts in order to profit from lemmatization and synonym normalization.

'totally wrong', we capture this as follows,

$$\texttt{confidence}(p) = \begin{cases} 1 - \eta \dfrac{\texttt{imp-score}(p)}{\theta + 1} & : \quad \texttt{select}(p) = \texttt{YES} \\ \min\left(1, \dfrac{\texttt{imp-score}(p) - \theta - 1}{\sigma - \theta - 1}\right) & : \quad \text{else} \end{cases}$$

where $0 \leq \theta < \sigma - 1$. The parameter $0 < \eta < 1 + \frac{1}{\theta}$ can be used to balance the YES scale of confidence scores with the NO scale of confidence scores.

2 Description of Runs

The two submitted runs both instantiate the general scheme for selecting valid hypothesis and for assigning confidence scores.

- In the recall-oriented first run, $\theta = 2$ mismatches ('imperfections') were allowed for YES decisions.
- In the more precision-oriented second run, $\theta = 1$ mismatch was allowed for YES decisions.

We chose $\sigma = 8$ in both runs, i.e. the confidence score for NO decisions will reach 1 if the imp-score detects 8 or more mismatches. It was intended to use $\eta = 1$ in all runs. However, the confidence scores in the submitted run #1 were based on $\eta = 1.2$ by mistake.

3 Evaluation

The viability of the basic approach of robust inferential entailment checking is witnessed by Table 1 which lists the results of MAVE in the answer validation exercise. 'Accuracy' measures the proportion of correct decision, and CWS is the confidence-weighted score.[3] Although the approach does not search for a minimal set of failed literals, but only skips literals according to a simple 'longest partial proof' strategy, the computed failed literal counts are obviously significant with respect to the validity of entailment: There is a consistent decrease in precision if more imperfections θ (usually skipped literals) are allowed, but also the intended boost in recall. The table further reveals that the two submitted runs (which admit one or two mismatches) represent the best compromise of all evaluation measures. Assuming provability of all literals ($\theta = 0$) is too restrictive in terms of recall, while admitting $\theta = 3$ or more mismatches considerably lowers precision.

MAVE features the use of question proofs for validating hypotheses, by comparing the answer part of the hypothesis with the lexical concepts in that snippet sentence which contributes most to the proof of the question. This mechanism is intended to increase recall when the hypothesis is syntactically ill-formed. In this case, no proof of the hypothesis is possible because no logical representation can be constructed. However, the question can usually be parsed so that a proof of the question can be tried. The hypothesis will then be compared with the answer determined by the proof of the question in a way which does not assume a parse of the hypothesis (i.e. only based on results of lexical analysis). To see the benefits of this technique, consider Table 2 which lists the results when no question-related indicators are used.[4] For $\theta = 1$, the recall decreases by 9.6% when no question-related information is available; for $\theta = 2$, the question

[3]For computing the confidence weighted score (CWS), $\sigma = 8$ and $\eta = 1$ were used throughout. Notice that for $\theta = 2$, exactly the same results are obtained for $\eta = 1$ and $\eta = 1.2$, i.e. the numbers shown for $\theta = 2$ indeed correspond to the submitted first run.

[4]i.e. when omitting the second argument of min in the definition of $\texttt{imp-score}(p)$.

θ (run)	Precision	Recall	F measure	Accuracy	CWS
0	0.8496	0.2720	0.4120	0.8051	0.8505
1 (run #2)	0.7238	0.3711	0.4906	0.8085	0.8457
2 (run #1)	0.5878	0.4929	0.5362	0.7859	0.8331
3	0.4880	0.5779	0.5292	0.7418	0.8134
4	0.4107	0.6771	0.5112	0.6750	0.7883
5	0.3658	0.7337	0.4882	0.6138	0.7638
6	0.3289	0.7705	0.4610	0.5477	0.7429

Table 1: Results of the MAVE system for the AVE-2006 test collection

θ	Precision	Recall	F measure	Accuracy	CWS
0	0.9452	0.1955	0.3239	0.7952	0.8418
1	0.8361	0.2748	0.4137	0.8044	0.8413
2	0.6882	0.3626	0.4750	0.7987	0.8375
3	0.5597	0.4646	0.5077	0.7738	0.8280
4	0.4508	0.5326	0.4883	0.7198	0.8138
5	0.3963	0.6062	0.4793	0.6693	0.7987
6	0.3615	0.6544	0.4657	0.6230	0.7832

Table 2: MAVE results based on hypothesis proofs only

indicators even contribute 13.0% of recall. Precision is lowered somewhat when using question data (by 11.2% for $\theta = 1$ and 10.0% for $\theta = 2$) but remains on a high level, and we get an overall improvement as witnessed by the higher f-measure and accuracy in the submitted runs. Table 3 shows that this increase in performance can only be achieved when both hypothesis and question indicators are combined. The table lists the results of MAVE when no information about hypothesis provability is used.[5] In fact, there is a clear decay in recall when deciding the validity of a hypothesis only based on the proof of the question and a subsequent matching with the hypothesis. Interestingly, the simplistic method of validating the hypothesis from a proof of the question and a comparison with the relevant snippet segment does not result in a loss of precision.

The definition of `imp-score`(p) also involves indicators (like `hypo-triviality`, `hypo-collapse` or `question-lexical-focus`) which serve to detect false positives. Table 4 shows the results obtained when omitting these indicators from the scoring function. Specifically, it demonstrates that the proposed indicators are effective in removing false positives, because precision drops by 14.4 percent when the

[5]i.e. when omitting the first argument of min in the definition of `imp-score`(p).

θ	Precision	Recall	F measure	Accuracy	CWS
0	0.8571	0.2380	0.3725	0.7987	0.8504
1	0.7468	0.3343	0.4618	0.8044	0.8481
2	0.6031	0.4476	0.5138	0.7873	0.8393
3	0.5081	0.5326	0.5201	0.7532	0.8244
4	0.4379	0.6487	0.5228	0.7027	0.8046
5	0.3910	0.7167	0.5060	0.6486	0.7822
6	0.3451	0.7450	0.4717	0.5811	0.7602

Table 3: MAVE results based on question proofs only

θ	**Precision**	**Recall**	**F measure**	**Accuracy**	**CWS**
0	0.7059	0.2720	0.3926	0.7888	0.8149
1	0.6000	0.3909	0.4734	0.7817	0.8052
2	0.5072	0.4986	0.5029	0.7525	0.7879
3	0.4319	0.5836	0.4964	0.7027	0.7650
4	0.3725	0.6827	0.4820	0.6136	0.7376
5	0.3390	0.7394	0.4648	0.5726	0.7128
6	0.3093	0.7790	0.4428	0.5078	0.6925

Table 4: MAVE results without false positive tests

false positive indicators are ignored. This effect is most marked when the recall base is small (most false positives like *'Gianni Versace is Gianni Versace'* are tautologies and thus provable for $\theta = 0$). Recall is not altered significantly, i.e. the indicators are sufficiently selective for removing mostly false examples.

4 Conclusion

We have presented the MAVE system for robust inferential answer validation in the MultiNet framework. Despite its small knowledge base, the system works reasonably well in the AVE exercise – mainly because most negative cases are strikingly false and easily recognized. Moreover, a relevant number of the positive cases need limited inference or are even trivial (i.e. constructed from a substring of the snippet). Thus AVE systems can be pretty simple-minded as long as QA systems use simple techniques, and they must become smarter only as the QA systems which generate the answers become smarter as well. Apart from that, the relative success of the MAVE prototype is explained by its robust proving strategy which tolerates a few mismatches and gaps in the coded knowledge. The most obvious improvement of MAVE will be the integration of large lexico-semantic resources like Germanet; this step is necessary to boost recall beyond 50%. Now that the annotated AVE 2006 corpus for German is available as a development collection, the design of improved scoring metrics and training of statistical classificators also become an option.

References

[1] Sven Hartrumpf. *Hybrid Disambiguation in Natural Language Analysis*. Der Andere Verlag, Osnabrück, Germany, 2003.

[2] Sven Hartrumpf, Hermann Helbig, Constantin Jenge, and Rainer Osswald. BenToWeb deliverable D6.2. Technical report, FernUniversität in Hagen, 2006.

[3] Sven Hartrumpf, Hermann Helbig, and Rainer Osswald. The semantically based computer lexicon HaGenLex – Structure and technological environment. *Traitement automatique des langues*, 44(2):81–105, 2003.

[4] Hermann Helbig. *Knowledge Representation and the Semantics of Natural Language*. Springer, Berlin, 2006.

[5] S. Marthen. Untersuchungen zur Assimilation größerer Wissensbestände aus textueller Information. Master's thesis, FernUniversität in Hagen, Hagen, Germany, 2002.