A Comprehensive Resource to Evaluate Complex Open Domain Question Answering

Silvia Quarteroni, Alessandro Moschitti

DISI - University of Trento
38050 Povo (Trento)
silviaq@disi.unitn.it, moschitt@disi.unitn.it

Abstract

We describe two corpora of question and answer pairs collected for complex, open-domain Question Answering (QA) to enable answer classification and re-ranking experiments. We deliver manually annotated answers to non-factoid questions from a QA system on both Web and TREC data. Moreover, we provide the same question/answer pairs in a rich data representation that includes syntactic parse trees and predicate argument structures and is compatible with the SVM-light toolkit. Experimenting with the above corpora allowed us to learn effective answer classifiers and re-rankers to improve the accuracy of our baseline QA system.

1. Introduction

Question Answering (QA) is a discipline that integrates Information Retrieval with Natural Language Processing technology in the purpose of finding accurate answers to natural language questions. While the first QA systems were conceived as natural language interfaces to small databases (Simmons, 1965), the discipline has evolved to encompass much wider information sources, scaling up to the Web (Kwok et al., 2001).

Most current QA systems can be defined as "open-domain" systems, as they aim at addressing questions of any type and concerning virtually any domain. Question types, or more appropriately expected answer types, are generally divided into two groups: factoid and non-factoid. The former group refers to answers that can be reduced to a fact, such as a name, geographical entity or date; in contrast, non-factoid or "complex" QA aims at finding definitions, descriptions, manners or reasons, and in general types of answers that go beyond a concise phrase.

1.1. Complex Question Answering

Non-factoid Question Answering is among the most complex and interesting problems in the natural language literature (Kazawa et al., 2001; Cui et al., 2005), as finding complex answers requires deep linguistic processing. However, there has been limited interest in specifically evaluating this type of application: TREC-10, the 2001 edition of the major QA evaluation campaign, remains to our knowledge the first of a limited number events where a large number of non-factoid questions was to be addressed by participant systems (Voorhees, 2001). The CLEF campaign also introduced 50 definition questions in the 2005 edition (Vallin et al., 2006), and has been dealing with an increasing number of complex question types in more recent years.

In this work, we focus on the types of complex questions falling into the coarse *Description* category of the question taxonomy designed to classify the TREC-10 test questions in (Li and Roth, 2002). This coarse-grained class mostly includes definitions, but also true descriptions, procedures (how- questions) and reasons (why-questions). According to (Li and Roth, 2002), 138 TREC-10 questions compose such a class; these are available as part of the UIUC corpus at: `http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/`.

In particular, this paper presents a complete resource to study the relations between such complex question types and their answers in an open-domain Question Answering system.

1.2. A resource to learn answer classifiers

In previous work (Quarteroni et al., 2007; Moschitti et al., 2007; Moschitti and Quarteroni, 2008), we have been confronted with the need to experiment with a number of machine learning models in order to classify and re-rank candidate answers to complex questions. Our models combined kernel functions applied on different relational representations of questions and answers: words, POS tags, syntactic parse trees and predicate argument structures.

In order to experiment with classifiers and re-rankers, we needed training and testing instances formed by complex questions and an ordered list of candidate answers from an existing Question Answering system for each of these To this end, we used YourQA (Quarteroni and Manandhar, 2009), our open-domain Question Answering system, designed to address both factoid and non-factoid questions and able to return answers alternatively from the Web or from a closed corpus.

The 138 complex questions in the UIUC corpus were submitted to YourQA and its top 20 answers were used to collect a corpus of candidate answers. The latter were manually labeled by two annotators according to their level of correctness with respect to their question.

Section 2. briefly describes YourQA's algorithm, while Section 3. introduces answer classification and re-ranking based on structural features and discriminative approaches. Section 4. describes the corpora collected from YourQA's answers to the UIUC description questions by retrieving documents from the Web and a TREC corpus, respectively. Finally, Section 5. summarizes experiments carried out using the corpora.

2. YourQA: Open-domain QA

YourQA (Quarteroni and Manandhar, 2009) is an open-domain, primarily Web-based Question Answering system. As most state-of-the-art systems (Kwok et al., 2001), YourQA is organized according to three phases: question processing, document retrieval and answer extraction.

During the first phase, the query is classified according to a taxonomy of factoid or non-factoid answer types; the two top expected answer types are estimated and the query is submitted to the underlying IR engine. Then, in the document retrieval phase, the top 20 documents found by the IR engine are retrieved and split into sentences.

Finally, during answer extraction, document sentences are compared to the question in the light of the expected answer types and candidate answers are selected; this phase requires additional details as it forms the baseline for any subsequent classification and re-ranking approaches.

2.1. Answer extraction

The answer extraction phase is centered on a sentence-level similarity metric applied to the query and to each retrieved document sentence to identify answers according to a combination of lexical, syntactic and semantic criteria.

In particular, based on the outcome of the question classifier, the answer extraction module determines whether the expected answer type belongs to the factoid group (in YourQA, the following factoids are defined: person, organization, location, quantity and time) or not.

In the first case, the required factoid is pinpointed down to the phrase or word level in each candidate answer sentence using factoid QA techniques, such as Named Entity recognizers and regular expressions.

In the case of non-factoid expected answer types, additional criteria are adopted to compute the similarity between the candidate answers and the original question: these match word $n-$grams, syntactic chunks, and phrase groups such as {head noun, verb, prepositional phrase} between the question and the answer. The final question-answer similarity metric therefore results from a weighted combination of the above similarity criteria.

2.2. Answer Format

Candidate answers are ordered by decreasing similarity (the IR engine rank of the answer source document is used as a tie-breaking criterion) and returned to the user surrounded by their original passage. While we here focus on the characteristics of YourQA's answers, full details about the system's answer extraction process and question/answer similarity metric are reported in (Quarteroni and Manandhar, 2009).

Figure 1 reports a snippet of YourQA's result format. The answer passage contains a sentence in boldface, corresponding to the document sentence obtaining the highest similarity score according to YourQA's answer extraction algorithm. This choice is due to the fact that the system is intended to provide a context to the exact answer; moreover, our focus on non-factoids made it reasonable to provide answers in the form of sentences.

3. Answer Re-ranking via Question/Answer Classification

State-of-the-art QA systems often perform a further step to answer extraction where additional, finer-grained criteria are employed to estimate the correctness of candidate answers; optionally this results in a re-ranked answer output. This phase is particularly useful for non-factoid expected answer types, where lexical features are often insufficient to provide accurate answers. Indeed, due to the small number of query keywords (often one), the number of common tokens between question and answer are not predictive of answer correctness.

Such a problem may be illustrated by considering the definition question $q =$ *What is autism?* and the two following answer candidates:

a_1 *Autism is a disease characterized by inability to relate to people.*

a_2 *Autism affects millions of people.*

Here, a lexical similarity metric such as the one described in Section 2. would give identical results when applied to (q, a_1) and (q, a_2); however, a_1 is clearly a much preferable answer. In these conditions, the use of answer classifiers and re-rankers working with structural text representations can highly contribute to understanding question/answer relations, and indeed what makes a good definition. This is illustrated in Section 3.1.

3.1. Structural representations

Since the last decade, a number of natural language processing approaches have been turning towards structural feature representation in the last decade (Zhang and Lee, 2003; Shen and Lapata, 2007).

Indeed, several tree-based feature representations have been explored within machine learning frameworks to study their impact on complex textual understanding tasks. Such representations include syntactic parse trees (PTs); for instance, Figure 2 reports a PT as output by the Charniak parser (Charniak, 2000).

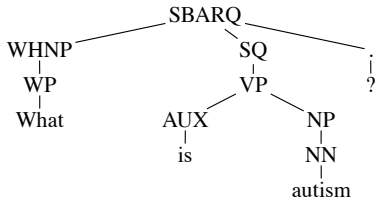

Figure 2: Syntactic parse tree of: *What is autism?*

Another example are Predicate-argument structures (PASs), that encode a more compact textual representation in terms of semantic roles; for instance, Figure 3 reports a PAS tree following PropBank semantics (Palmer et al., 2005).

3.2. Discriminative approaches based on structures

In the QA domain, tree kernels (Zhang and Lee, 2003) have proven to be effective in encoding syntactic parse trees in

1. Title: , URL: null, **Google Rank:** 1, **file:** 0_ AP880526-0070 _170334

An unfortunate mixup in the growth of the brain before birth may be responsible for autism, according to research published today. Researchers who made scans of victims' brains found that one structure buried deep within the head is often poorly developed. While this may not be the sole cause of autism, it could help explain many of the weird and stubborn symptoms of this illness. Dr Eric Courchesne and colleagues from Children's Hospital Research Center in San Diego pinpointed the abnormality in a region of the cerebellum, which lies at the base of the skull. This structure helps control movement, learning and some kinds of behavior.

Figure 1: Top answer extracted by YourQA from the AQUAINT corpus to the TREC 2001 question: "What is autism?"

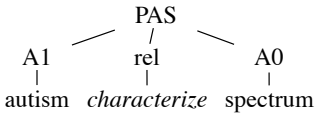

Figure 3: Compact Predicate Argument Structure of: *Autism is a characterized by a spectrum of disorders.*

learning algorithms exploiting to the robustness of Support Vector Machines (SVMs) to irrelevant features (Vapnik, 1995). More recent work, e.g. (Shen and Lapata, 2007) has shown that shallow semantic information in the form of predicate argument structures (PASs) improves the automatic detection of correct answers to a target question.

The intuition underlying the use of structural features for textual classification is that an overlap in the structure of two texts indicates a semantic relation, such as question/answer similarity. Consequently, Q/A similarity may be expressed as a function of common substructures in the trees representing the question and the answer.

Tree kernel functions are examples of such similarity functions; simply put, these receive as input a tree-based representation of the question and answer, enumerate both question and answer subtrees and then compute the number of matches between the subtrees. For instance, Figure 4 reports the syntactic parse tree of *Autism is a disease* as well as a number of its subtrees as enumerated by the syntactic tree kernel function defined in (Collins and Duffy, 2002).

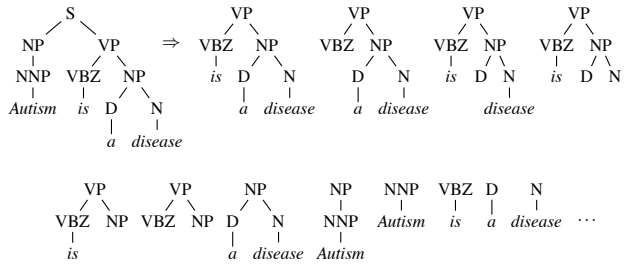

Figure 4: Parse tree of: *Autism is a disease* and some of its sub-trees as enumerated by the syntactic tree kernel.

In order to conduct answer classification and re-ranking experiments with kernel functions, a large amount of training instances, i.e. questions and candidate answers, is necessary. The two YourQA answer corpora collected for this purpose are discussed in Section 4.

4. The WEB-QA and TREC-QA Corpora

In order to obtain answers for our machine learning experiments, YourQA was deployed with two alternative IR engines during the document retrieval phase:

1. Google, to retrieve Web documents[1],
2. Lucene[2], to retrieve news articles from AQUAINT 6[3], the latest corpus released for TREC.

The resulting corpora, named WEB-QA and TREC-QA, contain 1309 and 2256 sentences respectively.

TREC-QA was necessary to align with the methodology followed by traditional QA system evaluation drawn from IR on a closed corpus. WEB-QA was particularly interesting to test the abilities of a fully Web-based open domain QA system, and to assess whether creating relational data representations based on the results of "off-the-shelf" parsers and semantic role labelers on Web data would yield effective learning algorithms.

Both corpora are available at: `disi.unitn.it/~silviaq/resources.html`; each corpus is delivered in the two following formats:

1. the judgment files resulting from the manual annotation of WEB-QA and TREC-QA,
2. the representation of annotated data as an input to SVM-light (i.e. training and testing files).

4.1. Judgment files

Judgment files contain a text version of YourQA's output, i.e. up to 20 answer paragraphs for each question. It is important to note that as the QA system does not necessarily select answers from each retrieved document and may discard unsuitable answer candidates, the number of paragraphs per question may vary.

In the judgment files, each sentence is manually annotated based on how well it answers the corresponding question. The annotation follows a Likert scale between 1 (totally incorrect) and 5 (completely correct). Two judges carried out the annotation task, reaching an inter-annotator agreement judged substantial (Cohen $\kappa = 0.635$).

[1] `google.com`
[2] `lucene.apache.org`
[3] available at: `trec.nist.gov/data/qa`

An excerpt of a judgment file is reported in Figure 5. Here, the top Web answer paragraph as returned by YourQA for the question *What is autism?* appears together with the original Google rank obtained by the question. The paragraph is composed of two sentences, the first one reaching the maximum judgment of 5, the second being classified as an incorrect answer (score 1). It can be noted that a Q/A similarity metric such as the one applied in YourQA's answer extraction phase would find it particularly difficult to distinguish between the above two sentences. Indeed, both share exactly one keyword with the question, "autism".

4.2. Training/testing files for SVM-light

We have been using the WEB-QA and TREC-QA corpora in a number of machine learning experiments to conduct complex answer classification and re-ranking. In particular, we tested the following kernel functions:

- linear kernels on words (BOW) and Part-of-Speech tags (POS),
- sequence kernels on words (WSK) and on Part-of-Speech tags (POS_{SK}),
- syntactic tree kernels (STK) on parse trees obtained via the Charniak parser (Charniak, 2000),
- shallow semantic tree kernels (PTK) on Predicate Argument Structures, obtained via the Semantic Role Labeling system described in (Moschitti et al., 2005).

Furthermore, we implemented combinations of the above kernels in the SVM-light-TK toolkit[4], that allows to design new functions in SVM-light (Joachims, 1999).
To experiment with these, we constructed five-fold cross validation splits containing training/testing instances derived from the judgment text files illustrated in Sec. 4.1.[5]
In particular, to simplify the classification task, we isolated for each paragraph in the judgment files the sentence with the maximal judgment and labeled it as $+1$ if its judgment was above 3 and -1 otherwise. For instance, given the question: *What are invertebrates?* the sentence: *At least 99% of all animal species are invertebrates* was labeled -1 , while the sentence: *Invertebrates are animals without backbones* was labeled $+1$.
Following this convention, WEB-QA contains 416 positive instances out of 1309 (31.8%), while TREC-QA contains 261 out of 2256 (11.6%): indeed, finding an answer to a question is simpler on the Web than on the smaller TREC corpus.
Each training/testing instance is a representation of a

$$< question, candidate\_answer\_sentence >$$

pair returned by YourQA. It is composed by a concatenation of the following information:

- A binary label ($+1$ or -1), indicating whether $candidate\_answer\_sentence$ has been judged as a correct answer to $question$ by the manual annotators;
- a unique identifier, composed by the concatenation `questionID:paragraphID:answerID`,
- 24 slots containing the following representations used by the kernel functions we defined:

  slots 0-4: Question parse tree (used by STK); BOW and POS as used by linear and sequence BOW/POS kernels[6]; BOW+POS; syntactic heads[7];

  slots 5-9: up to 5 question PAS_{PTK} (PAS used by PTK);

  slots 10-14: dummy slots;

  slots 15-18: Answer parse tree (used by STK), BOW and POS tags (used by linear and sequence BOW/POS kernels), BOW+POS, syntactic heads;

  slots 19-23: up to 5 answer PAS_{PTK} (used by PTK).

Figure 6 illustrates an example of an instance in the training/testing file format.

5. Classification & Re-ranking Experiments

To show the soundness and usefulness of our corpus for empirical studies, we briefly report our experiments using the representations in Section 4.2.

5.1. Answer classification

The objective of answer classification was to learn a binary answer classifier using the above training instances to determine whether candidate answers were correct answers to the corresponding questions. We tried several feature combinations by selecting different portions of the available information of the training instances and experimenting with the corresponding kernel functions.
Table 1 reports the accuracy over five folds achieved by different kernels on WEB-QA. We note that:

1. BOW achieves very high accuracy, comparable to the one produced by PT;
2. WSK improves on BOW, showing that word sequences are very relevant for this task;
3. the highest performing combination of features are PAS_{PTK} + WSK + BOW, further improving on BOW as a standalone.

A comparative analysis with the results obtained on TREC-QA, also in Table 1, suggests that:

1. the F1 of all models is lower than for WEB-QA, due to the fewer positive instances in the training corpus;
2. BOW denotes the lowest accuracy;
3. Sequence Kernels are beneficial, as POS_{SK} improves on POS (and PT);

[4] available at `dit.unitn.it/moschitti/`

[5] These are also available at: `disi.unitn.it/~silviaq/resources.html`.

[6] A slight modification of the STK applied to such tree representations implements the BOW/bag of POS feature

[7] Obtained following (Collins, 1999)

```
PARAGRAPH 1, GOOGLE RANK: 2
 Autism is a complex developmental disability that typically appears during the
 first three years of life and is the result of a neurological disorder that affects
 the normal functioning of the brain, impacting development
 in the areas of social interaction and communication skills.
 SENTENCE [2] (BEST) JUDGMENT: <5>
 Both children and adults with autism typically show difficulties in verbal and
 non-verbal communication, social interactions, and leisure or play activities.
 SENTENCE [3] JUDGMENT: <1>
```

Figure 5: A paragraph from the judgment file for the WEB-QA answers to the TREC 2001 question: "What is autism?"

```
+1 //binary label
103:16:57 //identifier with format QuestionID:AnswerParagraphID:sentenceID
//QUESTION (Q) FEATURES
|BT| (SBARQ (WHNP (WP What))(SQ (VP (AUX is)(NP (JJ mad)(NN cow)(NN disease))))(. ?)) //PT
|BT| (BOX (What *)(is *)(mad *)(cow *)(disease *)(? *)) //BOW
|BT| (BOX (WP *)(AUX *)(JJ *)(NN *)(NN *)(. *)) //POS
|BT| (BOX (WP What)(AUX is)(JJ mad)(NN cow)(NN disease)(. ?)) //BOW+POS
|BT| (BOX (SBARQ *)(WHNP *)(SQ *)(. *)(WP *)(VP *)) //SYNTACTIC HEADS
|BT| (PAS null) //up to 5 PAS (this one has none as the verb is ``to be'')
|BT| (PAS null)
|BT| (PAS null)
|BT| (PAS null)
|BT| (PAS null)
|BT| (TOP null) |BT| (TOP null) |BT| (TOP null) |BT| (TOP null) |BT| (TOP null) //dummy slots

//ANSWER (A) FEATURES
|BT| (S (NP (NP (`` ``)(JJ Mad)(NN cow)('' '')(NN disease))(, ,)(NP (NP (DT an)...  //PT
|BT| (BOX (`` *)(Mad *)(cow *)('' *)(disease *)(, *)(an *)(enigmatic *)(nervous *)(disorder *)
...  //BOW
|BT| (BOX (`` *)(JJ *)(NN *)('' *)(NN *)(, *)(DT *)(JJ *)(JJ *)(NN *) ...  //POS
|BT| (BOX (`` ``)(JJ Mad)(NN cow)('' '')(NN disease)(, ,)(DT an)(JJ enigmatic)(JJ nervous)
      (NN disorder) ...  //BOW+POS
|BT| (BOX (S *)(NP *)(VP *)(. *)(NP *)(, *)(NP *)(, *)(VP *)(CC *)(VP *)(`` *)(JJ *)(NN *)('' *)
...  //SYN HEADS
|BT| (PAS (A0 (disorder))(R-A0 (that))(rel kill)(A1 (thousands))(AM-LOC (britain)))  //up to 5 PAS
|BT| (PAS (A0 (disorder))(rel caus)(A1 (friction)))
|BT| (PAS (A0 (disorder))(rel threaten)(A1 (industry)))
|BT| (PAS null)
|BT| (PAS null)
|ET| //END OF INSTANCE
```

Figure 6: A labeled training instance for "What is mad cow disease?". Carriage returns and comments are introduced for clarity, however each training instance corresponds to only one row in the actual files.

4. Predicate Argument Structures add further information, as the best model is POS_{SK} + PT + PAS_{PTK}.

To relate our results to a reasonable baseline, we first measured the F1 of the answers corresponding to the top five documents returned by the IR engine and the top five answers as ranked by YourQA. Our results (Table 2) show that YourQA is slightly more accurate than its IR engine, and that our top Q/A classifiers greatly outperform YourQA.

Classifier F1	IR engine	YourQA	Top Q/A classifier
WEB-QA	35.9±4.0	36.8±3.6	**68.2**±4.3
TREC-QA	21.3±1.0	22.9±1.5	**39.1**±6.9

Table 2: F1 (± std.dev.) of the IR engine (Google resp. Lucene), of YourQA and of the top Q/A classifier on the WEB-QA and TREC-QA corpora

5.2. Answer re-ranking

Finally, Table 3 reports the Mean Reciprocal Rank value for the top 5 interpretations (MRR@5) as ranked by the IR engine and by YourQA's answer extractor, showing that the latter is much accurate in both WEB-QA and TREC-QA. Furthermore, when using the binary output of our top classifiers to rearrange YourQA's answers, we achieve a MRR of 81.1% on WEB-QA. On TREC-QA, the re-ranker produces a smaller improvement due to the higher complexity of the TREC dataset.

WEB-QA							
Model	BOW	POS	POS_{SK}	WSK	STK	PAS_{PTK}	PT+PAS_{PTK}+WSK
Classifier F1	65.3±2.9	56.8±0.8	62.5±2.3	65.7±6.0	65.1±3.9	50.8±1.2	**68.2**±4.3
TREC-QA							
Model	BOW	POS	POS_{SK}	WSK	PT	PAS_{PTK}	PT+PAS_{PTK}+POS_{SK}
Classifier F1	24.2±5.0	26.5±7.9	31.6±6.8	4.0±4.2	33.1±3.8	23.6±4.7	**39.1**±6.9

Table 1: Classification $F1 \pm$ std. dev. of several kernels on WEB-QA and TREC-QA

MRR@5	IR engine	YourQA	Top Q/A classifier
WEB-QA	49.0±3.8	56.2±3.2	81.1±2.1
TREC-QA	16.2±3.4	30.3±8.9	34.2±10.6

Table 3: MRR@5 (± std.dev.) of the IR engine (Google resp. Lucene), YourQA and the top Q/A classifier on WEB-QA resp. TREC-QA

6. Conclusions

Complex Question Answering involves a deep understanding of question/answer relations, such as those characterizing definition and procedural questions and their answers.
To contribute to the improvement of this technology, we deliver two question and answer corpora for complex questions, WEB-QA and TREC-QA, extracted by the same QA system, YourQA, from the Web and from the AQUAINT-6 data collection respectively. We believe that such corpora can be useful resources to address a type of QA that is far from being efficiently solved.
WEB-QA and TREC-QA are available in two formats: judgment files and training/testing files. Judgment files contain a ranked list of candidate answers to TREC-10 complex questions, extracted using YourQA as a baseline system and manually labelled according to a Likert scale from 1 (completely incorrect) to 5 (totally correct).
Training and testing files contain learning instances compatible with SVM-light (Joachims, 1999); these are useful for experimenting with shallow and complex structural features such as parse trees and semantic role labels.
Our experiments with the above corpora have allowed to prove that structured information representation is useful to improve the accuracy of complex QA systems and to rerank answers.

Acknowledgements

This research has been partially supported by the EC project "Trustworthy Eternal Systems via Evolving Software, Data and Knowledge" (EternalS, project number FP7 247758). We are grateful to Diego De Cao for his contribution to the document retrieval phase of our experiments.

7. References

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. NAACL*.

M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proc. ACL'02*.

M. Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

H. Cui, M. Kan, and T. Chua. 2005. Generic soft pattern models for definitional QA. In *Proc. SIGIR*. ACM.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods*.

H. Kazawa, H. Isozaki, and E. Maeda. 2001. NTT question answering system in TREC 2001. In *Proc. TREC*.

C. C. T. Kwok, O. Etzioni, and D. S. Weld. 2001. Scaling question answering to the web. In *Proc. WWW*.

X. Li and D. Roth. 2002. Learning question classifiers. In *Proc. ACL*.

A. Moschitti and S. Quarteroni. 2008. Kernels on linguistic structures for answer extraction. In *Proc. ACL*.

A. Moschitti, B. Coppola, A. Giuglea, and R. Basili. 2005. Hierarchical semantic role labeling. In *Proc. CoNLL 2005 shared task*.

A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proc. ACL*.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

S. Quarteroni and S. Manandhar. 2009. Designing an interactive open domain question answering system. *Natural Language Engineering*, 15(1):73–95.

S. Quarteroni, A. Moschitti, S. Manandhar, and R. Basili. 2007. Advanced structural representations for question classification and answer re-ranking. In *Proc. ECIR*.

D. Shen and M. Lapata. 2007. Using semantic roles to improve question answering. In *Proc. EMNLP-CoNLL*.

R. F. Simmons. 1965. Answering english questions by computer: a survey. *Comm. ACM*, 8(1):53–70.

A. Vallin, B. Magnini, D. Giampiccolo, L. Aunimo, C. Ayache, P. Osenova, A. Penas, M. de Rijke, B. Sacaleanu, D. Santos, and R. Sutcliffe. 2006. Overview of the clef 2005 multilingual question answering track. In *Accessing Multilingual Information Repositories*, pages 307 – 331.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

E. M. Voorhees. 2001. Overview of the TREC 2001 Question Answering Track. In *Proc. TREC*.

D. Zhang and W. Lee. 2003. Question classification using support vector machines. In *Proc. SIGIR'03*, Toronto, Canada. ACM.

Using High-Quality Resources in NLP: The *Valency Dictionary of English* as a Resource for Left-Associative Grammars

Thomas Proisl, Besim Kabashi

Friedrich-Alexander-Universität Erlangen-Nürnberg
Department Germanistik und Komparatistik
Professur für Computerlinguistik
Bismarckstr. 6, 91054 Erlangen
{tsproisl,kabashi}@linguistik.uni-erlangen.de

Abstract

In Natural Language Processing (NLP), the quality of a system depends to a great extent on the quality of the linguistic resources it uses. Due to the unpredictable character of valency properties, a reliable source for information about valency is important for syntactic and semantic analysis. With this in mind, we discuss how the *Valency Dictionary of English* in machine-readable form can be used as a resource for NLP. We will show that the valency data can be integrated into a Left-Associative Grammar and thus can be used for accurately parsing natural language with a rule-based approach.

1. Introduction

Valency as a model for the description of language goes back to Tesnière (1959) and his dependency grammar. The main idea of valency theory is that certain words, most prominently verbs but also members of other word classes such as adjectives or nouns, have a determining influence on the syntactic structure of a sentence.[1] The verb *admire*, for example, requires a noun phrase or a wh-clause expressing the thing or person admired to form a syntactically and semantically well-formed sentence:[2]

(1) a. He was not one for *admiring* the sunset. [BNC: J10 1822]
b. I know that Jack is a master chairmaker and I *admire* what he does. [BNC: A0X 292]

Valency dictionaries give detailed descriptions of these phenomena for every single word – this is necessary because even semantically related or similar words can differ in their valency properties.[3] It is these detailed descriptions that make valency dictionaries such valuable resources for NLP purposes.

Among the most well-known valency resources are COMLEX (Grishman et al., 1994), VALEX (Korhonen et al., 2006) and FrameNet (Fillmore et al., 2003). In this paper we want to show how valency information from the *Valency Dictionary of English* (Herbst et al., 2004) can be integrated into a Left-Associative Grammar.

[1]The notion of valency is also called complementation or subcategorization in other theoretical frameworks.

[2]Examples marked with the subscript 'BNC' have been extracted from the British National Corpus (2007), distributed by Oxford University Computing Services on behalf of the BNC Consortium. All rights in the texts cited are reserved.

[3]The nouns *amateur* and *master*, for example, require specific prepositions to express the theme of amateur- or mastership. There is, however, no way to exactly predict which noun requires which preposition (*master at sth.* vs. *amateur in sth.*).

2. The *Valency Dictionary of English*

2.1. The Printed Dictionary

The *Valency Dictionary of English* (VDE) is "a highly specialized dictionary that attempts to provide a detailed description of valency" (Herbst et al., 2004, vii). It contains valency descriptions of 511 verbs, 274 nouns and 544 adjectives.[4] Each entry presents the syntactic valency of the lemma in the form of formal patterns illustrated by authentic examples drawn from the *Bank of English*. The verb *discuss*, for example, possesses a valency pattern + wh-$CL_{P(it)}$ that indicates that *discuss* can take a wh-clause as complement (2-a). The subscript P(it) indicates that the wh-clause can occur as subject of a finite passive clause (2-b) and that when occurring as subject, extraposition with a dummy subject *it* is possible (2-c).

(2) a. Then they *discussed* what they had in mind for the publicity. [BNC: CHE 1968]
b. Why this should be so will be *discussed* in a moment [...]. [BNC: A75 922]
c. It has been *discussed* whether this increase may also be related to the proliferative capacity of the cells [...]. [BNC: FTE 726]

VDE entries also indicate the senses of the valent uses of a lemma, provide a list of all complements and the patterns in which they occur, group semantically similar complements together and informally characterize these groups of semantically similar complements. Let us give an example to illustrate the point.

(3) If you are living in council property you must *discuss* adapting your house with the housing department. [BNC: A0J 1891]

Example (3) is an instance of another valency pattern of *discuss*, namely + N_P/V-ing_P + with N. The VDE

[4]The VDE was created at the universities of Erlangen, Reading and Augsburg. The descriptions are based on the *Bank of English* and have been checked and complemented by native speaker informants (Herbst et al., 2004, xxxix).

tells us that the subject (which is not indicated in the pattern) belongs to the group of complements labelled I. The passivizable noun phrase (N_P) and its alternative, the passivizable clause introduced by the ing-form of a verb (V-ing$_P$), belong to the group of complements labelled II. The prepositional phrase introduced by *with* followed by a noun phrase (with N) belongs to the group of complements labelled III. The VDE gives the following informal characterization of these groups of complements: "**A person^{I} can discuss a matter^{II} with another person^{III}**, i. e. talk about it."
The VDE contains only 1 329 entries, but as the majority of lemmata are high-frequent, they have a surprisingly high token coverage. The VDE covers 12.36% of all noun tokens in the BNC, 19.30% of all adjective tokens and 77.36% of all verb tokens. If the verbs *be, do, have* and *go* are excluded because of their grammatical functions which are not covered by the VDE entries, the VDE still covers 65.32% of all verb tokens in the BNC.

2.2. Derived Electronic Resources

The VDE "shows considerable affinity with NLP, even though it was not conceived with the use by automatic systems in mind" (Heid, 2007, 378). Consequently, efforts have been made to turn the VDE into an electronic resource that can also serve NLP purposes.
Heid (2007, 374–375) reports a positive effect of VDE data extracted by Spohr (2004) on syntactic analysis coverage of a Lexical Functional Grammar for English and concludes that the VDE can "indeed serve as valency dictionary for a formal grammar."
Electronic versions of the VDE commissioned by Mouton de Gruyter and worked on by Proisl (2008) are the basis for the latest electronic version, which is available online via the *Erlangen Valency Pattern Bank*[5] (Herbst and Uhrig, 2009). The *Pattern Bank* is aimed at linguists working in the area of valency and argument structure constructions. Due to licensing issues, it contains mainly the syntactic valency patterns and omits most of the other information from the dictionary.

3. Left-Associative Grammar

Left-Associative Grammar (LAG) is the formalism of the SLIM theory of language (Hausser, 2001).[6] In contrast to other grammar formalisms, LAG models natural language not by a hierarchy of substitutions but by a sequence of continuations.[7] To illustrate this in a simplified form, consider example sentence (4-a) which will be processed as indicated in (4-b).

(4) a. He opened the envelope. [BNC: GUF 2069]

b. ((((He + opened) + the) + envelope) + .)

In the first step, the first two word forms are concatenated by combination rules that make use of the lexical and grammatical categories of the word forms. The result contains the concatenated word forms and a so called rule package that indicates the possible continuations. Processing continues word form by word form until the last word form is reached. Formally, an LAG rule i can be represented as follows:

$$r_i\colon\ \mathrm{cat}_1\ \mathrm{cat}_2 \Rightarrow \mathrm{cat}_3\ \mathrm{rp}_i$$

The formalism of LAG is currently being used within Database Semantics (DBS) (Hausser, 2006) which is based on the SLIM theory of language.
LAG has proven adequate for modelling even complex phenomena of natural languages, e. g. the doubling or substitution of objects by pronominal clitics in Albanian (Kabashi, 2007).

4. Integration

We will illustrate the integration of the valency data available via the *Erlangen Valency Pattern Bank* into a Left-Associative Grammar by analyzing the following example:[8]

(5) The following analogy may prove helpful in understanding these statements. [BNC: HSE 176]

First, consider the relevant lexical information:

```
[sur: the
 cat: {(sn' snp) (pn' pnp) ...}]

[sur: following
 cat: {(adj)}  ]

[sur: analogy
 cat: {(sn) (for_npo' sn)
       (with_npo' sn) ...}]

[sur: may
 cat: {(nps' inf' v) ...}]

[sur: prove
 cat: {(adjp' inf)
       (npo' to_npo' inf) ...}]

[sur: helpful
 cat: {(to_npo' adj)
       (in_v-ing' adj) ...}]

[sur: in
 cat: {(npo' in_npo)
       (v-ing' in_v-ing) ...}]

[sur: understanding
 cat: {(npo' v-ing)
       (npo' as_npo' v-ing) ...}]

[sur: this
 cat: {(sn' snp) ...}]
```

[5] http://www.patternbank.uni-erlangen.de

[6] The letters of the acronym SLIM indicate the main principles of this theory of language: Surface Compositional Linear Internal Matching.

[7] According to the CoNSyx hypothesis put forward by Hausser (2001, 236), "[t]he natural languages are contained in the class of C1-languages and parse in linear time." The formalism of LAG is therefore regarded as particularly well-suited for NLP purposes.

[8] The use of resources in Natural Language Generation (NLG) is part of an ongoing project by B. Kabashi.

```
[sur: statement
 cat: {(sn) (of_npo' sn)
       (about_npo' sn) ...}]

[sur: .
 cat: {(ip)}]
```

The shortened[9] entries displayed here are feature structures containing the surface of a word form as a string and its grammatical categories as a set of tuples. The categories consist of segments indicating valency slots (marked with an apostrophe) and a segment expressing formal properties of the word form. The valency properties of *analogy, prove, helpful, understanding* and *statement* are based on information from the *Pattern Bank*,[10] the other entries were created by the authors.
In the first step, the determiner *the* (categorized, inter alia, as a singular noun phrase still needing a singular noun:[11] `(sn' snp)`) is combined with the adjective *following* by means of general linguistic rules. In the next step, the singular noun *analogy* fills the empty `sn'` valency slot of the determiner, thus eliminating the other category sequences of *the*. The next word form, the modal verb *may*, is categorized as needing both an NP in subjective case (`nps'`) and an infinitive (`inf'`). The category value `snp` is (by definition of variable restrictions) able to fill the valency slot `nps'` of the modal verb. The combination with the modal verb also rules out all the categorizations of *analogy* except the avalent one, `(sn)`.
In the next combination step, the lexical verb *prove* is able to fill the `inf'` slot of *may*. *Prove* possesses various valency patterns, inter alia one which demands an adjective phrase (`adjp'`). The variable restrictions allow that the next word, the adjective *helpful*, can fill this valency slot. All other category sequences of *prove* are ruled out by this combination step.
Helpful itself also possesses valency patterns. Indicated in the listing above are patterns demanding a prepositional phrase (PP) with *to* followed by an NP in objective case (`to_npo'`) or a PP with *in* followed by the ing-form of a verb (`in_v-ing'`). The next word form, the preposition *in*, illustrates how we are dealing with PPs. One of its category sequences is `(v-ing' in_v-ing)`. This means that *in* is able to fill the `in_v-ing'` valency slot of *helpful* but still has an open `v-ing'` slot.
By now, the mechanism should have become clear. In the next step, *understanding* fills the empty `v-ing'` slot of *in* and opens other slots. The demonstrative *this* fills the `npo'` slots of understanding but still needs a singular noun. *Statement* is able to fill this slot. The next word is a full stop. At this point, there are analyses that still have unfilled valency slots, e. g. the `as_npo'` slot of the second pattern of *understanding* or the PP slots of some patterns of *statement*. As the full stop ends the current sentence, these analyses are considered ungrammatical. However, there is also one analysis which has no open valency slots (using the first patterns of *understanding* and *statement*). The dependency structure of this analysis is visualized in the stemma in Fig. 1.

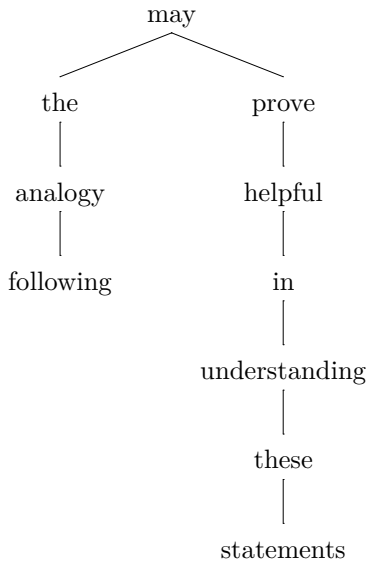

Figure 1: Stemma for example (5)

During the analysis, we have the possibility of fusing certain function words with content words (Hausser, 2006, 87–90). The resulting more compact dependency structure is shown in Fig. 2.

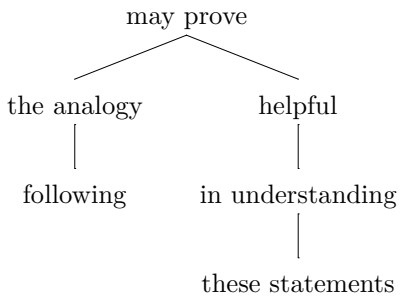

Figure 2: Stemma for example (5), illustrating the fusion of function words with content words

5. Problems

Here we will highlight only some of the more prominent difficulties, for a more detailed discussion cf. Spohr (2004, 34–40) and Proisl (2008, 107–110).
There are some pattern elements in the VDE that are not determined in their form by the valency carrier. The symbol ADV, for example, represents a valency slot that can be filled by an adverb phrase, a noun phrase, a prepositional phrase or an adverbial clause. This kind of underspecified category is difficult to deal with in an NLP system.
Sometimes, the VDE provides information on collocations or contextual lexical tendencies, e. g. + wh-CL

[9] We display only the attributes discussed in the text.
[10] Changes include the addition of valency slots for subjects of finite verb forms and explicit case marking for NPs.
[11] The alert reader will have noticed that the determiner is categorized as needing a noun, not vice versa. So, strictly speaking, we are dealing here with determiner phrases (DPs) instead of noun phrases (NPs).

(often: how or what) or [cannot] + N. Such information, although very valuable, is also difficult to integrate.
A further issue, which is not a "problem" as such, is coverage. While we pointed out above that token coverage is surprisingly high, the absolute number of entries is still quite small. It would therefore be desirable to increase the size of the resource.

6. Comparison

The *Valency Dictionary of English* and the treatment of valency in Left-Associative Grammar have been compared to other resources and formalisms (Proisl, 2008, 33–42, 50–68). We will give a very brief summary of the most important findings here.
A comparison with COMLEX, VALEX and FrameNet suggests that the VDE entries provide a more detailed (and in some cases also more accurate) description of the syntactic and semantic valency properties of their lemmata. As it is a highly specialized dictionary, the VDE can of course not compete with the wealth of non-valency information contained for example in COMLEX.
From a valency point of view, Left-Associative Grammar has two main advantages over other systems such as Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) or various flavors of Categorial Grammar, e. g. Dependency Categorial Grammar (Pickering and Barry, 1993). First, long distance dependencies can be treated in an uncontroversial way. Second, in contrast to other formalisms, LAG is closer in spirit to most versions of valency theory in that it treats valency exclusively as a property of words and not also as a property of constituents or phrases.

7. Conclusion

In this paper we tried to show three things. First, that the *Valency Dictionary of English* in electronic form is very well suited for being used as a resource in Natural Language Processing. Second, that the integration of its main data into the formalism of Left-Associative Grammar can be accomplished without problems. Third, that LAG provides a simple way of handling natural language phenomena.
A resource of high quality, such as the VDE, simplifies the grammar development because the developer need not be concerned with its correctness but can use it as a reliable source. Therefore, the use of such a resource makes it possible to derive automatic analyses of equally high quality.

8. References

The British National Corpus. 2007. Version 3 (BNC XML edition). Distributed by Oxford University Computing Services on behalf of the BNC Consortium. `http://www.natcorp.ox.ac.uk`.

Charles J. Fillmore, Christopher R. Johnson, and Miriam R. L. Petruck. 2003. Background to framenet. *International Journal of Lexicography*, 16(3):235–250.

Ralph Grishman, Catherine Macleod, and Adam Meyers. 1994. Comlex syntax: Building a computational lexicon. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, volume 1, pages 268–272.

Roland Hausser. 2001. *Foundations of Computational Linguistics: Human-Computer Communication in Natural Language*. Springer, Berlin, New York, 2nd edition.

Roland Hausser. 2006. *A Computational Model of Natural Language Communication: Interpretation, Inference, and Production in Database Semantics*. Springer, Berlin, Heidelberg, New York.

Ulrich Heid. 2007. Valency data for natural language processing: What can the *Valency Dictionary of English* provide? In Thomas Herbst and Katrin Götz-Votteler, editors, *Valency. Theoretical, Descriptive and Cognitive Issues*, pages 365–382. Mouton de Gruyter, Berlin, New York.

Thomas Herbst and Peter Uhrig. 2009. Erlangen Valency Pattern Bank – a corpus-based research tool for work on valency and argument structure constructions. Website. `http://www.patternbank.uni-erlangen.de`.

Thomas Herbst, David Heath, Ian F. Roe, and Dieter Götz. 2004. *A Valency Dictionary of English: A Corpus-Based Analysis of the Complementation Patterns of English Verbs, Nouns and Adjectives*. Mouton de Gruyter, Berlin, New York.

Besim Kabashi. 2007. Pronominal clitics and valency in Albanian: A computational linguistics perspective and modelling within the LAG-framework. In Thomas Herbst and Katrin Götz-Votteler, editors, *Valency. Theoretical, Descriptive and Cognitive Issues*, pages 339–352. Mouton de Gruyter, Berlin, New York.

Anna Korhonen, Yuval Krymolowski, and Ted Briscoe. 2006. A large subcategorization lexicon for natural language processing applications. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 1015–1020, Genua.

Martin Pickering and Guy Barry. 1993. Dependency categorial grammar and coordination. *Linguistics*, 31:855–902.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.

Thomas Proisl. 2008. Integration von Valenzdaten in die grammatische Analyse unter Verwendung des *Valency Dictionary of English*. Master's thesis, Philosophische Fakultät und Fachbereich Theologie, Friedrich-Alexander-Universität Erlangen-Nürnberg.

Dennis Spohr. 2004. Using *A Valency Dictionary of English* to enhance the lexicon of an English LFG grammar. M. A. study. Institut für maschinelle Sprachverarbeitung, Universität Stuttgart.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Klincksieck, Paris.