Translating news to CycL using the XLE parser

Janez Starc, Blaž Fortuna

Artificial Intelligence Laboratory
Jožef Stefan Institute
Jamova 39, SI-1000 Ljubljana
{janez.starc, blaz.fortuna}@ijs.si

Abstract

We built a pipeline for translating text into logical representation, which falls in the category of Machine Reading (MR). The essential component of the pipeline and our main contribution is the non-probabilistic rule-based framework. Other components are: syntatic parser, XLE, to extract gramamtical features from text, Enrycher, for additional natural language processing, and Cyc for semantic resources, reasoning, and its language CycL to respresent the translated knowledge. We defined and implemented several rules on the framework. We evaluated them on business news. In the discussion we identified several challenges in MR.

Prevajanje novic v jezik CycL s pomočjo razčlenjevalnika XLE

Zgradili smo cevovod za prevajanje besdila v logično predstavitev. Naše delo spada v področje strojnega branja. Glavna komponenta cevovoda in naš glavni prispevek je neprobabilistično programsko ogrodje, ki temelji na pravilih. Ostale komponente so: XLE – sintaktični razčlenjevalnik, Enrycher – storitev za procesiranje naravenega jezika in Cyc – semantični vir ter avtomatski pojasnjevalnik. Za predstavitev prevedenega znanja smo uporabili jezik CycL. Na programskem ogrodju smo definirali in implementirali nekaj pravil. Ocenili smo, kako prevajajo besedila iz poslovnih novic. V zaključku smo odkrili številne izzive v strojnem branju.

1. Introduction

Vast amounts of text are published online every minute. People all over the world are communicating through Facebook updates, Tweets, blogs, etc., or more formal discourses like news articles, academic papers, books, etc. The tendency is to extract as much information as possible from these sources and express this information in a logical representation. This data can be used to populate a particular knowledge base. Consequently, new knowledge can be inferred from the knowledge base.

We built a non-probabilistic rule-based system, which does several tasks sequentially. In the beginning, it obtains English textual data from a news article stream, IJS Newsfeed. Then it processes the data with natural language processing tools XLE parser (Maxwell and Kaplan, 1996) and Enrycher (Štajner et al., 2010). In the next step, the system translates the processed text into a logic language of Cyc system (Lenat, 1995), CycL. These logical statements are then asserted into the knowledge base of Cyc, CycKB. The main contribution of our work is a framework for implementing rules, which translate the output of the XLE parser, f-structures, to CycL. We named this component the Translator. We also defined, implemented and evaluated a few rules. Our system exploits external resources to semantically enrich its output.

Our paper falls into category of Machine Reading (MR). A study of requirements for MR has been done by (Clark and Thompson, 2009). Similar work to ours was done in (Witbrock et al., 2004) using different parsers. Also similar to our work is the approach taken by researchers at Parc (Crouch, 2005) (Crouch and King, 2006) (Bobrow et al., 2007). Many others took the semi-supervised approach, for e.g. (Etzioni, 2007) (Carlson et al., 2010). In (Ghosh et al., 2011) Markov Logic Networks (MLNs) were used for MR.

The rest of this paper is organized in the following way. We describe all third-party components, especially parts that we use in our system in Section 2. We present the workflow of our system in Section 3. In Section 4., we evaluate our system. We finish with conclusions and proposals for future work in the last section.

2. Description of components

2.1. IJS Newsfeed

To obtain online news we have used software IJS newsfeed[1]. It periodically crawls a list of RRS feeds and a subset of Google News to obtain links to news articles. Then it downloads and parses the articles to extract cleartext version of the article body, which can be further processed by Enrycher. This data is then available on two streams, cleartext and Enrycher processed. We captured news from the stream that is processed by Enrycher, which we present in the next subsection.

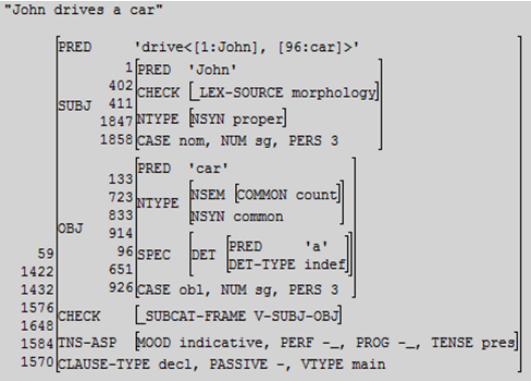
2.2. Enrycher

Enrycher[2] (Štajner et al., 2010) is a service that provides shallow and deep text processing at the document level. Main features of the system include topic and keyword detection, named entity extraction, word sense disambiguation, triplet extraction. We have used sentence splitting, topic detection, named entity resolution, co-reference and anaphora resolution in our system.

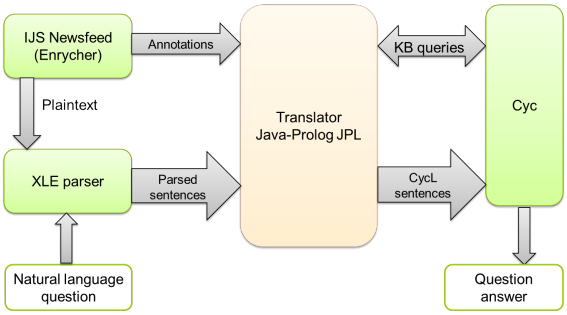
2.3. XLE parser

This software[3] (Maxwell and Kaplan, 1996) parses text with Lexical Functional Grammars (LFGs). It is a part of larger platform, XLE, which also provides rich graphical user interface for writing and debugging LFGs and facility

[1] `newsfeed.ijs.si`
[2] `http://enrycher.ijs.si`
[3] `www.parc.com`



Slika 1: F-structure



Slika 2: The Translator Pipeline

for generating text from LFGs. The main part of the platform is ordered rewrite rule system, which was used to produce semantic representations from the syntactic output of the parser (Crouch and King, 2006). This work has similarities to ours. We both use XLE parser, but different system to generate semantic representations.

The XLE parser produces two mutually constraining types of output, c-structures and f-structure. C-structures or structures of syntactic constituents are constituency trees. The f-structure analysis, on the other hand, treats the sentence as being composed of attributes, which include features such as number and tense, or functional units such as subject, predicate, or object. For example, f-structure of the sentence *"John drives a car."* is depicted on Figure 1. This sentence has only one f-structure. Since the system does not use any semantics there can be more than one solution to one sentence, and the parser produces a packed representation of all possible solutions. Using a form of Optimality Theory (Prince and Smolensky, 2004) solutions are ranked, based on ordered violable constraints. We use the highest ranked solution, which is written in Prolog format, for further processing.

2.4. Cyc

The last third-party component that we use is Cyc[4] (Lenat, 1995). This system includes CycKB, which is an ontology and knowledge base of every day common sense knowledge. The knowledge base contains nearly 500.000 terms, about 15.000 types of relations, and about 5 million assertions. The knowledge is represented by a formal language CycL. The CycL representation of the sentence *"John drives a car."* is presented on Listing 1.

Listing 1: CycL sentence

```
(#$thereExists ?ACTION
  (#$thereExists ?CAR
    (#$and
      (#$isa ?ACTION
        #$TransportInvolvingADriver)
      (#$isa ?CAR #$Automobile)
      (#$vehicle ?ACTION ?CAR)
      (#$driverActor ?ACTION #$John)
)))
```

[4] www.cyc.com

Cyc concepts start with *#$* and Cyc variables start with *?*. CycL sentences are split into microtheories. Each microtheory must not contain any contradictions. Furthermore, Cyc offers an inference engine, which derives answers on the queries using the knowledge base. It is also used to reject assertions that would be inconsistent with the knowledge base. We use Cyc to assert the translated sentence into its knowledge base. We use its inference engine to answer the CycL queries of the corresponding interrogative sentences. We also utilize data about subcategorization frames, which are stored in CycKB. There are two versions of Cyc technology available: the open source OpenCyc and the more complete version, ResearchCyc, which we use in our work.

3. The Workflow

In this section, we will present how our system works. The workflow is shown on Figure 2.

3.1. From news stream to f-structures

In the first step, we listen to IJS newsfeed stream for some time to download a sample of articles together with additional information produced by Enrycher. Each article is tagged with several topics keywords from SIOC ontology. We built a filter, which passes through articles that have been tagged with the selected topic. Using Enrycher's sentence and token splitting feature, another filter was built to filter out either too long or too short sentences. These types of sentences can be ungrammatical or the possibility of correct parse is very low. The parameters for this filter are minimal and maximal number of tokens in a sentence. All the filtered sentences are concatenated, and gathered in the plaintext file. This file is then sent as input to the XLE parser. The parser produces one Prolog file for each sentence. Each file includes several attributes including the actual sentence, metadata, the most probable f-structure, and c-structure.

3.2. Semantic resources from Enrycher

Enrycher annotates every named instance with its type. The following types are resolved: person, location, organization, date, percentage, amount of money. For every named instance we generated one corresponding Prolog compound term, which is available to Translator. One term has four arguments: the sentence, in which the instance appears, the mention of the instance, the name of the instance,

and the Cyc concept, which corresponds to the type of the instance. Here is an example of a compound term:

```
ne('She died in 1957 at age 90.','She','
    Mary',p_Person).
```

We can notice that there was a person named Mary mentioned in a sentence preceding the observed sentence. The author of the discourse replaced the word *Mary* with the pronoun *she* in the observed sentence. In this example, we exploited two Enrychers utilities: co-reference resolution and named entity resolution.

3.3. Semantic resources from Cyc

In the previous subsections, we described the first part of input to Translator: the actual sentence, its f-structure and information about its named entities. Now, we will present two types of semantic resources from Cyc that Translator uses. First, Translator can obtain all Cyc concepts denoted by a particular word. For instance, we would like to get all Cyc concepts, which correspond to word *plays* in the sentence *"John plays."*. The XLE parser provides shallow level linguistic information: this word is a verb and its lemma is play. For the pair *(play, verb)* Cyc returns the following list of concepts:

- `#$Playing`
- `#$PlayingAMusicalInstrument`
- `#$PlayingAGame`
- `#$PlayingAnAudioRecordedObject`

In addition, we also queried phrases for dentotaions. In this case, we did not add any linguistic information about the phrase.

The other type of semantic resources we utilize are *verbSemTrans* relations. This is the definition of the relation taken from CycKB:

> *(verbSemTrans VERB NUM FRAME TRANS)* means that the translation of word sense number NUM of VERB, appearing with subcategorization frame FRAME, is TRANS.

Example of such relation is presented on Listing 2. Subcategorization frame is type of sentence according to number and type of syntactic arguments that co-occur with the verb. For example, *Transitive noun phrase frame* corresponds to sentences that have subject and object. These two arguments must be noun phrases. Other examples of subcategorization frames are *Intransitive verb frame*, *Ditransitive noun phrase frame*, *Middle voice vrame*, etc. Last argument, the translation, is a CycL sentence that may include free variables like *:ACTION*, *:SUBJECT*, *:OBJECT*, *:INDIRECT-OBJECT*. These are later bound to corresponding CycL constructs.

Listing 2: Example of verbSemTrans relation

```
(#$verbSemTrans #$Drive-TheWord 0
  #$TransitiveNPFrame
    (#$and
      (#$isa :ACTION #
          $TransportInvolvingADriver)
      (#$vehicle :ACTION :OBJECT)
      (#$driverActor :ACTION :SUBJECT)))
```

The verbSemTrans relation from Listing 2 has been used to translate sentence *"John drives a car."* into CycL on Listing 1. Predicates *vehicle* and *driverActor* introduce semantics. Predicate vehicle requires *:OBJECT* to be instance of Cyc concept *#$TransportationDevice-Vehicle*. Analogously, predicate driverActor requires *:SUBJECT* to be instance of *#$Person*. Two interesting things can happen when trying to assert CycL sentence that is induced by this relation into CycKB. If one argument cannot fulfil the semantic requirements, e.g. *:SUBJECT* is not a person, then the whole sentence cannot be asserted into the knowledge base. Otherwise, if Cyc cannot prove that one of the arguments meets the requirements, e.g. cannot prove that *:SUBJECT* is a *#$Person*, than the whole sentence is asserted and *:SUBJECT* becomes a *#$Person*. Therefore, with the help of inference engine Cyc can also learn knowledge that was not explicit in the natural language sentence.

3.4. The Translator

In this subsection, we describe the main part of the Translator. We implemented this part of the system in Java and Prolog. We have chosen Prolog because one of the possible output format of the XLE parser is Prolog. The second reason is that the target language CycL is very similar to Prolog. To translate from Prolog to CycL and vice versa we developed a simple regex-based procedure. Because we could not make external calls to Cyc out of Prolog, we also had to use Java. To connect Java and Prolog we used JPL[5]. This is a bidirectional interface, which enables Prolog applications to exploit any Java classes, methods, instances, etc. Analogously, it allows any Java application to manipulate any Standard Prolog libraries, predicates, etc.

The main job of the Translator is to recursively execute the ordered rewrite rules. Rewrite rules replace the antecedent expression with the consequent expression if the antecedent expression complies with the rule. In our case, f-structures are replaced with CycL sentences. Rules are ordered in a sequence. The first rule that satisfies the conditions is executed and no other rules are tested or executed. One rule can have multiple outputs. Therefore, final output can consist of multiple CycL sentences. Translator executes the rules recursively. It starts with the whole sentence, then it recursively translates smaller parts of sentences, e.g. noun phrases, verb phrases. The smallest translated constituent is a word.

One of the main contributions of this paper is the implementation of the sequence of ordered rewrite rules, which is presented below.

1. ***Rule for verbSemTrans relations from Cyc.*** From the obtained f-structure of a sentence we can determine the subcategorization frame of the sentence. For example, the value of the feature *_SUBCAT-FRAME* in the f-structure on Figure 1 is *V-SUBJ-OBJ* (verb, subject, object), which corresponds to Transitive noun phrase frame. This frame together with the verb *drive* corresponds to verbSemTrans relation on Listing 2. The last argument of this relation is the incomplete

[5] `http://www.swi-prolog.org/packages/jpl/`

CycL of the underlying translated sentence. It is incomplete because terms like *:SUBJECT* still need to be instantiated. We implemented the following subcategorization frames:

- Transitive noun phrase frame (subject, verb, object), e.g., *"John drives a car."*
- Intrasitive verb frame (subject, verb), e.g., *"John swims."*
- Ditransitive prepositional phrase frame (subject, verb, object, preposition, oblique-object), e.g., *"John passed the ball to Tom."*
- Copula frame (subject, complement), e.g., *"John is sad."*

2. ***Rule for semantically poorer verbSemTrans relations.*** Frame and verb combinations that are not covered by the first rule, may be covered by this rule. This rule also uses verbSemTrans relations. But these are not stored in Cyc, instead, they are automatically generated. This is an example of such relation:

```
(#$verbSemTrans #$Drive-TheWord 1
  #$TransitiveNPFrame
  (#$and
    (#$isa :ACTION
       TransportInvolvingADriver)
    (#$actor :ACTION :OBJECT)
    (#$actor :ACTION :SUBJECT)
))
```

This relation is similar, but the CycL in the last argument is semantically poorer, than the one on Listing 2. This is because predicate *actors* means just that the last argument of the relation is somehow involved in the action.

3. ***Oblique-object rule.*** This rule is executed only if ditransitive prepositional phrase frame rule was applied before. It is used to correctly extract oblique objects from the prepositional phrase.

4. ***Free variable rule.*** This rule is executed on interrogative pronouns, e.g., who, what, in the interrogative sentences, which are recognized by *CLAUSE-TYPE - INT* attribute-value pair in the f-structure. Interrogative pronouns are recognized by *PRON-TYPE - INT* pair and are translated into CycL variables. Consequently, the final construct of the translation is not a CycL sentence, but a CycL query.

5. ***Cyc denotation rule.*** This rule finds a Cyc concept for a part of the sentence (see Subsection 3.3.)

6. ***Enrycher annotation rule.*** Details of resolving the annotation are presented in Subsection 3.2.. Names and types of entities are connected by this rule. Here is an example of the result of the rule:

```
(#$thereExists ?OBJECT
    (#$and
      (#$isa ?OBJECT #$Person)
      (#$nameString ?OBJECT
        "Tiger Woods")
)).
```

7. ***Noun phrase splitting rule.*** This rule splits the noun phrase into the main noun and the possible modifiers. Each modifier is connected to the main noun with the predicate *featureOf*, which is semantically very poor. For example, *"white rabbit"* is translated into `(#$featureOf #$Rabbit #$WhiteColor)`. Example of the semantically richer relation would be, `(#$mainColorOf #$Rabbit #$WhiteColor)`.

8. ***String instance rule.*** This is the last rule. It always succeeds. It is applied on a part or even whole sentences. It uses Cyc function InstanceFn. The result of the function is a Cyc concept, which is named after the functions only argument. Here is an example of the result of the rule:

```
(#$InstanceFn "quasicrystal")
```

3.5. Asserting sentences to Cyc KB

The result of the translation is one or, due to homonomy, multiple CycL sentences. For each natural language sentence, we create a microtheory, for example,

```
(#$MtWithFocalContentSentenceFn "Francisco
    Liriano will start for the Twins.")
```

We did not use a single microtheory for all senteces, because contradictions may appear due to news writing bias. We try to assert all possible translations of a particular sentence into its microtheory one by one. If a particular translated sentence is contradictory, it is rejected.

3.6. Question answering

The question answering feature was also added to our system. It is possible to make a natural language question. This question is translated to the CycL query. Cyc will answer the query by providing all the answers to it. If we asserted the translation of the sentence: *Clint Eastwood eats a steak.* We can ask a question like: *What does the actor eat?* This is the query translated from the sentence:

```
(#$thereExists ?SUBJECT
  (#$and
    (#$isa ?SUBJECT #$Actor)
    (#$thereExists ?ACTION
      (#$and
        (#$isa ?ACTION #$EatingEvent)
        (#$performedBy ?ACTION ?SUBJECT)
        (#$consumedObject ?ACTION ?OBJECT)
))))
```

The answer to the query, in which *?OBJECT* is the variable, is *(SKF-1534975054)*. This concept is a Skolem term that represents the particular steak that Clint Eastwood eats.

4. Evaluation

In this section, we will first present the empirical evaluation, and then an experiment that we conducted on our system. Since the start of this project, we have taken into account that translating text to knowledge representation is a hard problem, if not impossible. A system like ours would need a huge number of rules to get a good coverage. Since the basic unit of our input is a sentence, we filtered out sentences that are not going to produce good translations

without parsing. One type of such sentences are the ones that have a big probability of being incorrectly parsed by the syntactic parser. From our experience, these are either longer sentences, which consist of multiple clauses, or the ones that contain non-alphabetic characters, e.g., parenthesis, dashes, etc. Of course, there are also sentences that are grammatically incorrect, but these are hard to recognize without parsing. During evaluation, we also noticed that there is huge spread of quality of the evaluated sentences. Some of them are correctly translated, but they lack semantic richness. On the other hand, some translations are semantically very rich, but not all phrases are correctly translated.

In this paragraph, we will present an experiment that we conducted. We selected a controlled set of sentences from the IJS newsfeed stream, processed them, translated them to CycL, asserted them to CycKB and manually evaluated a fraction of translations. We only took articles from the business domain, because we expected text from this domain less complicated. To exclude other articles we used the SIOC tags from Enrycher. After obtaining the eligible articles, we split them into sentences. We retained the sentences that have 7 - 15 words, start with the capital letter, end with the period, and do not have any other characters than alphabetic characters, periods or currency signs. We got 19443 sentences from a total of 40624 articles. From these sentences we extracted 12245 mentions of named entities. We randomly selected 1000 sentences. These were then translated to CycL, and the translations were asserted to Cyc. Out of this set, 326 sentences had at least one valid Cyc translation. A valid translation is a CycL sentence, which is not necessary without contradictions. 204 sentences had at least one assertion. Out of these, 101 sentences had only one assertion; the others had ambiguous translations. One sentence had a maximum of 210 asserted translations. The average number of asserted translations was 3.0.

Of the sentences that had assertible translations, we randomly selected 50 sentences for human evaluation. One human evaluator observed three things on each sentence: the quality of the XLE f-structures (see Table 1), qualities of phrase denotations (see Table 2), the quality of the structure and semantic relations (see Table 3). Because word sense disambiguation was not applied in our system, there are multiple possible assertible translations. The evaluator made word sense disambiguation himself and has chosen the correct translation, and evaluated it. If some less important parts of the sentence, like adverbs in the beginning of the sentence, were not translated, the translation was not penalized.

No. of points	Description of quality class
3	The parse is completely correct.
2	The parse is almost correct. One part of the parse is not correct. However, it is good enough to be further translated.
1	The parse is wrong and it is not used for further processing.

Tabela 1: Scoring of f-structure quality

Annotation	Description of the denotation class
G (good)	The phrase is correctly denoted by a Cyc concept.
M (missing)	The phrase is encapsulated by InstanceFn function. Cyc should have a concept denoting this phrase.
P (poor)	The phrase is encapsulated by InstanceFn function. This phrase should be further split into smaller denotable units and Cyc should not have a concept denoting this phrase.
W (wrong)	The phrase is incorrectly denoted by a Cyc concept.
E (Enrycher)	The phrase denoted with Enrycher named entity resolution.

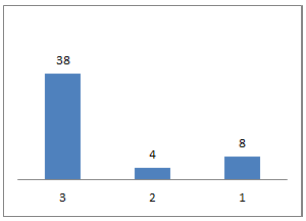
Tabela 2: Phrase denotation quality scoring

No. of points	Description of quality class
4	The structure is good and semantically rich.
3	The structure is good. However, some predicates have no sematic meaning.
2	Something in the structure is wrong.
1	The structure is completely wrong.

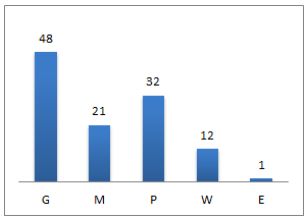
Tabela 3: Scoring of the quality of the structure and sematic relations

The results of the human evaluation are presented on Figure 3. The evaluation showed that 78% of the sentences were correctly parsed (Figure 3a). However, we should not judge the overall precision of the parser based on this number, because in this sample there are only sentences that have valid translation. Phrase denotation quality shows that about 42% of phrases have a corresponding Cyc concept (Figure 3b). There is only one phrase denoted with the help of Enrycher. This number is very low because Cyc denotations have priority over Enrycher denotations. The semantic and structure qualities are quite evenly distributed (Figure 3c). Sentences that had the lowest parsing score were automatically given the lowest structure score.

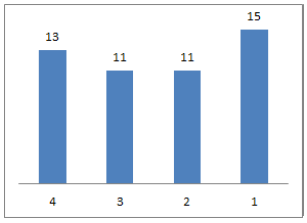
To analyse execution time to translate one sentence, we have to divide the processed into three parts: pre-processing part, XLE parsing and the translation. The whole dataset is processed in each step. These steps are not parallelized. The preprocessing part does not include the time that IJS newsfeed spends to download the articles and annotate them with Enrycher. About 30 articles are preprocessed in one second. XLE parsing of one sentence takes about a quarter of a second. However, we made this measurement on short sentences, which we used in the experiment. It took substantially more time, if we parsed sentences that were longer than the ones in the experiment. The translation and assertion time heavily depend on the number of ambiguous translations. The more translations that one sentence has the more time it takes to assert them to Cyc.



(a) F-structure quality



(b) Phrase denotation quality



(c) Structure and semantic quality

Slika 3: Quality distributions

However, in average two sentences can be translated and asserted in one second.

5. Conclusion

We have made many constraints in the process of translating and evaluation to get the precision that reflects on Figure 3. Therefore, our system is not complete enough to translate large amounts of news articles and then reason on the translated data. It turned out that our system is very useful to identify the problems that arise in text to logic translation. In contrary to many other systems, our system is non-probabilistic, recursive and rule based. Therefore, Prolog and JPL were very suitable for the job. Because our system is sequential, the drawback is that error propagates through the workflow. Evidence of this is also seen on Figure 3.

The XLE parser and its f-structures proved to be very useful in this kind of translation. Although, the diverse nature of the news language is not suitable for the parser. Its accuracy, especially for the longer sentences, is not acceptable. In addition, many sentences were grammatically questionable.

On the other hand, Cyc has a large ontology covering most of the phrases. However, ontology should be supplemented with additional concepts to improve to accuracy and semantic richness of the translation systems. The translation patterns stored in CycKB were manually created. We used the ones for verbs. It would be interesting to implement patterns for nouns, adverbs, etc. in our system. Nevertheless, there are not enough patterns to cover a language like news media language. This raises the question, whether is it possible to automatically create such patterns.

Our system also needs the mechanism for word sense disambiguation. Many systems learn from corpora that are annotated with word senses from a particular database. We believe that corpora for Cyc concepts do not exist yet. Therefore, a different kind of word sense disambiguation solution must be implemented.

Acknowledgements

This work was supported by Slovenian Research Agency and the ICT Programme of the EC under XLike (ICT-STREP-288342).

We would like to thank Tomaž Erjavec for useful comments and for the review.

6. References

D.G. Bobrow, B. Cheslow, C. Condoravdi, L. Karttunen, T.H. King, R. Nairn, V. Paiva, C. Price, , and A. Zaenen. 2007. Parcs bridge and question answering system. In *Proceedings of the GEAF 2007 Workshop*.

A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka, Jr., and T. M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 101–110, New York, NY, USA. ACM.

P. Clark and J. Thompson. 2009. A study of machine reading from multiple texts. In *Proceedings of AAAI Spring Symposium on Learning by Reading and Learning to Read*.

D. Crouch and T.H. King. 2006. Semantics via f-structure rewriting. *Proceedings of LFG06*, pages 145–165.

R. Crouch. 2005. Packed rewriting for mapping semantics to kr. In *Proceedings of the 6th International Workshop on Computational Semantics*, pages 103–14. Citeseer.

O. Etzioni. 2007. Machine reading of web text. In *Proceedings of the 4th international conference on Knowledge capture*, K-CAP '07, pages 1–4, New York, NY, USA. ACM.

S. Ghosh, N. Shankar, and S. Owre. 2011. Machine reading using markov logic networks for collective probabilistic inference. In *Appearing in the Proceedings of the European Conference on Machine Learning (ECML) Workshop on Collective Learning and Inference from structured data (CoLISD)*.

D.B. Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

J. Maxwell and R. Kaplan. 1996. An efficient parser for lfg. In *Proceedings of LFG*, volume 96, page 131.

A. Prince and P. Smolensky. 2004. *Optimality Theory: Constraint interaction in generative grammar*. Wiley Online Library.

T. Štajner, D. Rusu, L. Dali, B. Fortuna, D. Mladenić, and M. Grobelnik. 2010. A service oriented framework for natural language text enrichment. *Informatica (Ljublj*, 34(3):307–313.

M. Witbrock, K. Panton, S.L. Reed, D. Schneider, B. Aldag, M. Reimers, and S. Bertolo. 2004. Automated owl annotation assisted by a large knowledge base. In *Workshop Notes of the 2004 Workshop on Knowledge Markup and Semantic Annotation at the 3rd International Semantic Web Conference ISWC2004*, pages 71–80.