Ontology-based information extraction in agents' hands

Brigitte Endres-Niggemeyer

University of Applied Sciences and Arts
Faculty III - Media, Information and Design
Expo Plaza 12, 30539 Hannover, Germany
phone +49 511 92 96 - 2641
fax +49 511 92 96 - 26 03
Brigitte.Endres-Niggemeyer@fh-hannover.de

Abstract. This paper briefly reports on an agent team doing ontology-based information extraction (OBIE, IE) for summarization in clinical Bone Marrow Transplantation (BMT). The SummIt-BMT agents contribute to OBIE through their flexible use of ontological knowledge. They assess input text passages from web retrieval with respect to a user query. They use an ontology that supports IE in particular with concepts, propositions, unifiers and paraphrases. Sentences with IE hits are annotated with the IDs of ontology propositions that recognize an instance of their content in the sentence. The agents are beginners, but they perform. Distributing ontology-based IE to agents has some promise: it enables parallel processing, it eases tracking of decisions and their explanation to users.

1. An agent team for ontology-based information extraction

Imagine a team of agents who specialize in ontology-based information extraction for summarization (more detail in Endres-Niggemeyer et al. 2006, Endres-Niggemeyer 1998). Figure 1 presents them in their communication environment. For ease of use, the agents answer to simple German forenames. Their family names are derived from their function, sometimes with some influence of their structure or history. Currently there are, in the order of appearance:

Peter Question
Kurt DummyIRBean
Frieda TextToPropMini
Heini DispatchProposition
Hugo SpotOntoProps
Rudi VerifyPropArguments
Herta CheckPropRelation
Paula SumUpHits

The agent community distributes summarization and IE tasks as observed in competent humans: proceed step-by-step and apply all available resources at a time. Every agent roughly performs a strategy as seen in human summarizers.

The agents are Java classes that extend the jade.core.Agent[1]. They run in a JADE container and use standard ACL (Agent Communication Language)[2] means of interaction. All agents share a set of simple calls. Most calls consist of the name of the addressed agent and a simple German codeword: *los* (go), *mehr* (more), *fertig* (done). Only the tidy-up agent *Paula* is also assigned a more sophisticated command when she has to reorganize results for presentation: *sumup* (sum up). When broadcasting the close-down message to all agents, *Kurt* says *schluss* (finish) to make the agents delete.

The system blackboards serve data communication. The *ScenarioBoard* stores the query specification and the findings of the agents. While they interpret a sentence, the agents exchange data via the *TextBlackBoard*. External input comes from the text passage retrieval result. At the end of a session, the retrieval result (organized in documents, paragraphs and sentences) is augmented with the agents' relevance judgements. They mark the relevant text clips, which are presented to the user.

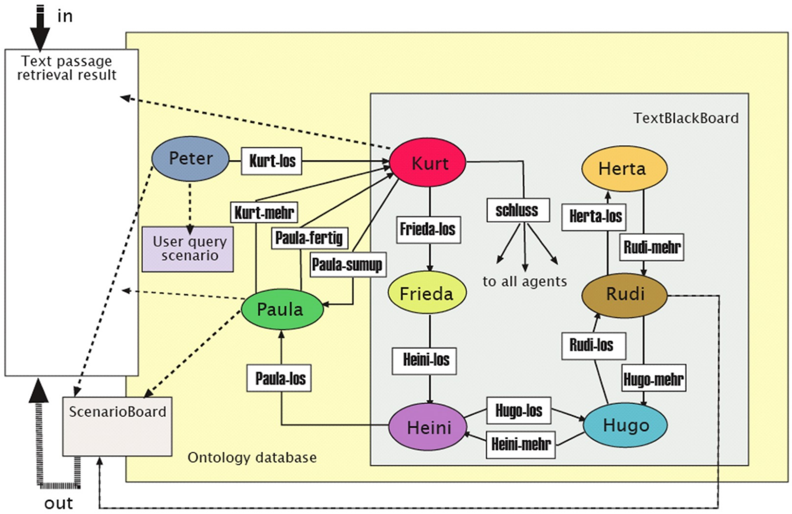

Fig. 2. The agents and their communication network. Dotted arcs represent data input/output

2. The Ontology

The agents and the system users share an ontology of the application domain Bone Marrow Transplantation (BMT). The ontology was developed by human experts from a corpus of US BMT papers and user queries of physicians at Hanover Medical School. It is stored in a MySQL[3] database. In the ontology the agents find the types of knowledge they need for IE (see Table 1): concepts, propositions, proposition syntax records, unifiers, paraphrases, scenarios, and some technical help tables.

We use a Prolog style first order predicate logic representation. Inside the MySQL database, all knowledge items are split into separate tables. Propositions

1 See JADE at http://jade.tilab.com/
2 http://www.fipa.org/repository/aclspecs.html
3 http://www.mysql.de/

comprise a head and a set of arguments allocated to the propositionhead and propositionargument tables, respectively. Their proposition ID keeps them together. Every proposition obeys a syntax record that states its argument roles. Syntax tables are built like proposition tables. Unifiers are lists of concepts provided by domain experts. They unify ontology propositions and text-based candidate propositions: a concept of the accredited unifier adapts the ontology proposition so that it matches a candidate proposition from input. This expands the coverage of the ontology propositions. Paraphrases map ontology propositions to possible surface formulations. They are *macropropositions* (Kintsch and van Dijk 1983): parsed surface phrases with argument roles as variables, so that one paraphrase can serve a class of proposition occurrences in text. The scenario representation stores the whole presentation on the JSP[4]-based user interface.

Table 1. Ontology database overview.

Knowledge unit	Quantity	Database tables
concept	4813	concept, conceptsynonym, hyperconcept
japanese concept	4683	multilanguage, japan
proposition	5054	propositionhead, propositionargument, signature
syntax	507	syntaxhead, syntaxargument, predicate, predsyn
unifier	680	unifier, unifcalc
paraphrase	11845	paraphrasehead, paratoken, parapropidlist
scenario	61	scenario, scenfamily_hr, scenarioblock, scenarioblocklist, scenariofield, scenariofieldPI, scenariofieldPIlist, scenariofieldlist, scenarioquery, scenblockoption, scenqueryword, scenquestionargument

3. The agents' jobs

The agents specialize in different IE subtasks. They produce a summarization effect by extracting only propositions that match the query and by throwing away doubles. All agents activate each other as often as needed.

Scenario interpretation. *Peter* accepts a user query scenario and the user's start signal. Into the scenario form, the user has entered what is known about the current situation and what knowledge is missing. The agent parses this organized query, deposits the resulting propositions on the *ScenarioBoard* and activates *Kurt*.

Table 2. Ontology propositions' hits for the demo sentence.

No.	ID	Wording (FOL)
1	17650	administer (, patient, ganciclovir, intravenous)
2	17652	administer (, patient, ganciclovir, intravenous, low dose, short-course)
3	17656	administer (, patient, ganciclovir)
4	17685	administer (, patient, methotrexate)
5	21054	haveRiskClass (patient, low risk, disease progression)
6	21055	haveRiskClass (patient, high risk, cytogenetic risk)
7	21056	haveRiskClass (patient, high risk, chromosome aberration)
8	21057	haveRiskClass (patient, high risk, age)
9	22097	prevent (patient, broad-spectrum antibiotic, , antimicrobial prophylaxis, posttransplantation)

Input. *Kurt* fetches the query and obtains results from outside web retrieval and text passage retrieval. He submits good input sentences one by one to the parser (the

4 Java Server Pages - http://java.sun.com/products/jsp/

Connexor[5] FDG parser) and feeds wording and parser output into the agents' production line by putting it onto the *TextBlackBoard*. He calls *Frieda*.

Let us assume for the sake of a demo that *Kurt* comes up with the sentence

"All patients at cmv risk were administered high-dose ganciclovir."

It will be hit by 9 ontology propositions (see table 2). We follow proposition 17685.

Candidate propositions in a parsed sentence. *Frieda* picks up the new input. She finds candidate propositions in a parsed sentence and annotates them (see table 3, columns 8 – 13). She distinguishes verbal, prepositional and attributive candidate propositions. As soon as her annotation is done, *Frieda* activates the agent *Heini.*

Table 3. An example dependency parser output with 3 annotated propositions of different types. Columns 1 - 7 display the parse, columns 8 – 13 the proposition candidates.

1	2	3	4	5	6	7	8 9		10 11		12 13	
1		all	DET	@DN> %>N	det:		*					
2		patient	N NOM PL	@SUBJ %NH	subj:		arg1		modhead2			
3		at	PREP	@<NOM %N<	mod:		praep1		praep 2			
4		cmv	ABBR NOM SG	@A> %>N	attr:		arg1		modarg 2		attr 3	
5		risk	N NOM SG	@<P %NH	pcomp:		arg1		modarg 2		attrbase 3	
6		be	V PAST PL	@+FAUXV %AUX	v-ch:		aux					
7		administer	EN	@-FMAINV %VP	main:		pred1					
8		high-dose	A ABS	@A> %>N	attr:		arg1					
9		ganciclovir	N NOM SG	@OBJ %NH	obj:		arg1					

Sending propositions to interpretation. *Heini* is the proposition dispatcher. He selects the propositions one by one from the current sentence and initiates their verification. As long as he has input, he submits it to *Hugo*. When all propositions of the current sentence are done, he calls *Paula*, the tidy-up agent.

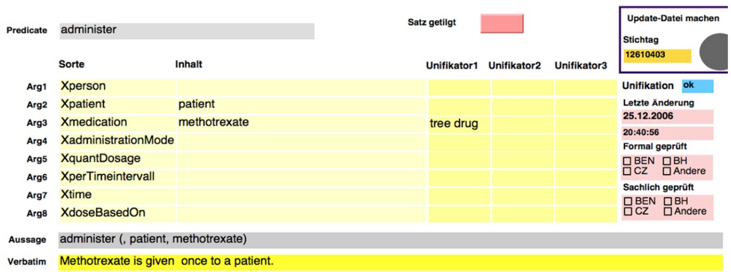

Fig. 2. An ontology proposition equipped with a unifier for all drugs.

Finding ontology propositions. *Hugo* checks whether the current proposition shares at least two ontology concepts with any of the ontology propositions. As soon as he is done with a concept pair and has some results, *Hugo* passes the text-based proposition with the IDs of selected ontology propositions - and possibly some add-ons due to unifiers - to the *TextBlackbord.* He activates *Rudi*. If *Hugo* cannot find matching ontology propositions, he returns to *Heini* and asks for new supplies.

When *Hugo* begins to treat a new proposition, he puts the IDs of occurring ontology

5 http://www.connexor.com/

concept IDs into its record. Using them he may find several concept pairs that call ontology propositions. Eventually he selects proposition 17685 (see figure 2):

administer (, patient, methotrexate).

His pick takes the direct and the unifier pathway. According to the unifier in the proposition, any drug of the ontology may be put in. As *ganciclovir* is needed, *Hugo* adds an ersatz argument that contains *ganciclovir*. He puts his results into the package for *Rudi*.

Concept subsumption. *Rudi* tries to subsume text-based propositions under ontology propositions with at least two concepts in agreement. If the subsumption works, the proposition from text may be a legitimate instance of the subsuming ontology proposition, as far as ontology concepts are concerned. If so, *Rudi* passes it to *Herta*. She will inspect the verbal relation.

When *Rudi* fetches proposition 17685 that *Hugo* proposed, he looks for ersatz arguments, finds one and puts it in. Now his version of proposition 17685 says:

administer (, patient, ganciclovir).

As figure 2 shows, the proposition has some open slots. *Rudi* tries to fill them, subsuming concepts from the text-based proposition. He succeeds once: he subsumes *high dose* under concept 43174 (*quantity qualifier*) in position 5. Now his proposition reads:

administer (, patient, ganciclovir, , high dose).

As all obligatory arguments are satisfied, *Rudi* passes his result (see table 4) to *Herta*.

Table 4. Concept-based IE result.

pos	propid	concept	cid	role	hyper -cid	unif	required	testable	match
0	17685	administer	0	pred	0	0	false	false	false
1	17685		0	0	41884	0	false	true	false
2	17685	patient	38811	arg	38811	0	false	true	true
3	17685	ganciclovir	39204	arg	39375	1418	true	true	true
4	17685		0	0	42583	0	false	false	false
5	17685	high dose	43174	arg	39595	0	false	false	true

Verification of the verbal relation. *Herta's* task is to check the verbal tie that keeps the ontology concepts together. She verifies the relation information against sets of paraphrases. If a paraphrase provides a relation wording that is compatible with the ontology proposition chosen by *Rudi* and the relation wording of the text surface, *Herta* has found the last missing link. She states the recognition success for that proposition by assigning the ID of the subsuming ontology proposition to the text sentence. She asks *Rudi* for fresh input.

In the test case, *Herta* receives *Rudi's* reworked ontology proposition 17685. She writes the ontology concept's IDs into the parse of proposition 1 (cf. table 3).

Herta procures her paraphrase set of proposition 17685. She will find the test paraphrase 14068437 (see table 5) that will fit. *Herta* seizes the concepts found in the text-based proposition via their hypercids and attaches them to the hypercids / argument roles of the paraphrase. Then she checks in three passes:

From satisfied roles she goes towards the root of the dependency tree and checks all items on her way to ok.

She compares the verbal chain of the proposition and paraphrase. There should be a reasonable fit, depending on word classes. If so, *Herta* places her controls.

At the end, *Herta* starts from the ontology proposition arguments without fillers. Again she goes up the dependency hierarchy and sets all words on her way to optional.

If *Herta* obtains all ticks as needed, she has verified the verbal relation. In the present case, she has found

"patient is administered ganciclovir".

She writes the hit ID to the TextBB. *Paula* will reorganize all results.

Table 5. Paraphrase 14068437 of proposition 17685. The dependency relation is noted in *word ID* and *dep-target*. Relation type is declared in *depend relation*.

para-no phrase ID	word ID	token	word class morphology	syntactic function	depend relation	hyper-cid	dep-target
140684371	2	Xpatient	N NOM SG	@SUBJ %NH	subj:	38811	3
140684372	3	be	V PRES SG3	@+FAUX %AUX	v-ch:	0	4
140684373	4	administer	EN	@-FMAINV %VP	main:	0	1
140684374	5	Xmedication	N NOM SG	@OBJ %NH	obj:	39375	4

Cleaning up. *Paula* is the organizer. When processing of a sentence is finished and has brought some results, *Paula* stores the sentence with the recognition results to the *ScenarioBoard*. She tells *Kurt* to provide new input. When all input is done, *Paula* reorganizes the *ScenarioBoard*. She sorts the recognition IDs of individual sentences so that she obtains orderly recognition profiles. Based on the profiles and the wording of the sentences, *Paula* weeds out doubles. Surviving hits are added to their text clips in the retrieval result. *Paula* asks *Kurt* to close down the agent community.

4. Evaluation

Table 6. Final overall scores of the agents. R1 is the agents' first run, R2 the second one.

Abstract	number of sentences	R1 sentence hits	R1 mean raw score	R2 sentence hits	R2 mean raw score	mean final rating
Bcr-abl1	13	9	2.05	11	1.37	3.7
Bcr-abl2	7	2	3.67	6	2.0	2.2
Bcr-abl3	10	2	4.23	5	1.75	2.2
Childhood ALL1	8	4	3.77	5	2.63	3.2
Childhood ALL2	10	3	4.0	2	1.78	1.5
Childhood ALL3	16	6	3.18	10	2.39	3.6
CMVganciclovir1	12	6	4.35	6	2.15	2.7
CMVganciclovir2	12	8	2.25	7	2.0	3.6
CMVganciclovir3	12	1	5.0	6	2.4	3.0

A biochemist and the author evaluated the agents' performance in a testbed with a small sample of Medline abstracts. Methods were adapted from qualitative field research. The agents ran twice. Between their two runs, the judges improved the ontology, and results became much better. Ontology quality matters. Often the agents stumble over simple human errors, sloppy categorizations or into ontology gaps. In overcrowded areas, they are obstructed by too many chances to derive the same recognition result.

In their second run, the agents achieved fair scores. They are still beginners, but they come up with results. Table 6 shows their marks on a familiar 5-score scale.

5. Sources and related approaches

SummIt-BMT integrates knowledge from many sources. Ontology and agents are based on empirical observation of human summarizers (Endres-Niggemeyer 1998), following human task organization as much as possible. Humans summarize content. A domain terminology / an ontology is a natural start for their IE activities. For IE (Appelt and Israel 1999) and summarization an extended ontology is required, so propositions, unifiers, paraphrases and scenarios were integrated. The agents' IE is adaptive (Turmo et al. 2006), given a domain ontology. It seemed consistent to distribute the human-like strategies to an agent community (JADE - Bellemine et al. 2007) and to give the agents task-specific blackboards for data interchange and storage (already in the SimSum system – Endres-Niggemeyer 1998). Implementing this at the state of the art led to OBIE, to agents using blackboards, to unifier use, to paraphrases incorporating parsed macropropositions. As mainstream evaluation does not work for the agents-and-ontology approach, a small-scale evaluation procedure was drawn from qualitative field research methods (Glaser and Strauss 1980).

6. Conclusion

Ontology-based IE (for summarization) can be distributed to an agent team. This has advantages: Agents' decisions can be tracked more easily. The agents may explain them. New agents are easily integrated, so that the community "learns". If running in parallel, agent teams may be fast and scale up well.

References

Appelt, D., Israel, D.: Introduction to Information Extraction Technology. Tutorial at the International Joint Conference on Artificial Intelligence (IJCAI-99), Stockholm (1999), http://www.dfki.de/~neumann/esslli04/reader/overview/IJCAI99.pdf

Bellifemine, F. L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. Wiley (2007)

Endres-Niggemeyer, B.; Jauris-Heipke, S; Pinsky, M.; Ulbricht, U.: Wissen gewinnen durch Wissen: Ontologiebasierte Informationsextraktion. Information - Wissenschaft & Praxis, 301-308 (2006), http://endres-niggemeyer.fh-hannover.de/OntologiebasierteIE.pdf

Endres-Niggemeyer, B.: Summarizing information. Springer, Berlin (1998)

Glaser, B.G., Strauss, A.L.: The discovery of grounded theory: Strategies for qualitative research. 11th ed. Aldine Atherton, New York (1980)

Kintsch, W., van Dijk, T. A.: Strategies of discourse comprehension. Academic Press, New York (1983)

Turmo, J., Ageno, A., Català, N.: Adaptive information extraction. ACM Computing Surveys 38, 2, Article 4 (2006), http://www.lsi.upc.es/~ncatala/home-angles.html