Using Dialogue Corpora to Extend Information Extraction Patterns for Natural Language Understanding of Dialogue

Roberta Catizone, Alexiei Dingli, Robert Gaizauskas

Department of Computer Science

University of Sheffield

E-mail: roberta@dcs.shef.ac.uk, alexiei@dingli.org, r.gaizauskas@dcs.sef.ac.uk

Abstract

This paper examines how Natural Language Process (NLP) resources and online dialogue corpora can be used to extend coverage of Information Extraction (IE) templates in a Spoken Dialogue system. IE templates are used as part of a Natural Language Understanding module for identifying meaning in a user utterance. The use of NLP tools in Dialogue systems is a difficult task given 1) spoken dialogue is often not well-formed and 2) there is a serious lack of dialogue data. In spite of that, we have devised a method for extending IE patterns using standard NLP tools and available dialogue corpora found on the web. In this paper, we explain our method which includes using a set of NLP modules developed using GATE (a General Architecture for Text Engineering), as well as a general purpose editing tool that we built to facilitate the IE rule creation process. Lastly, we present directions for future work in this area.

1. Information Extraction for Dialogue

Why use Information Extraction for Dialogue? Information Extraction techniques for extracting meaning are generally applied to text documents, for example newspaper reports, scientific papers, or blogs, rather than to transcribed spoken dialogues. However, we have chosen to apply IE to dialogue for the following reason: Dialogue utterances tend not to be well-formed sentences, yet convey meaning to the hearer. Since utterances are not well-formed, a full-parsing method is not as desirable as a pattern matching approach with shallow syntactic parsing to identify NPs and VPs. This lends itself to an IE template-based approach. We devised our method when developing a demonstrator for a dialogue system in the domain of Office chat (for the EU-funded Companions project), but it could be applied to any dialogue domains.

In our system, IE patterns are part of a Natural Language Understanding module (Figure 1). While all inter-module communication in the overall system takes place via a blackboard, this module in effect takes input from the upstream Speech Recognition and Dialogue Act tagging modules for the current user utterance and from the downstream Dialogue Manager for the system response to the previous utterance. It outputs a shallow meaning representation for the current user utterance which is passed on to the Dialogue Manager for formulating a response to the user.

For example, one sort of input our system must handle in the domain of Office Chat is utterances that express the user's emotional attitude about their day or project or task. Such utterances may be conceived of as ATTITUDE relations between a person and a day, project or task, with subtypes WORRY, HAPPY, ANNOY, etc. The WORRY Relation is signaled by words such as 'worry', 'be worried', 'be troubled', 'be concerned' and 'be afraid' and so on. Relations also have attributes which, in our domain, are attitude-type (WORRY, HAPPY, etc), attitude-subject (Person) and attitude-object (Person, Project or Task).

Given this framework, the NLU output for the sentence "**I'm a bit worried**." is:

<entity id=e1 type=person user=yes>I</entity>'m a bit
<relation-signal id=rs1 type=attitude
subtype=worry>worried</relation-signal>
<relation id=R1 type=attitude subtype=worry
attitude-subject=e1 relation-signal=rs1>

which may be expressed more abstractly in a logical form representation as:

object(user:person),
object (e1:Person),
attribute (e1, user, true),
object(r1:attitude),
attribute(r1,subtype,worry)
attribute(r1,attitude-subject, e1)
attribute(r1,attitude-object,unknown)

Such representations, in which both entities and relations are reified, are convenient given partial information, which may result either from imperfect analysis or from information being distributed across multiple sentences.

To derive such meaning representations automatically we use entity and relation extraction techniques. To develop entity and relation extractors one can pursue either a manually authored rule-based approach or a machine learning approach. The advantages and disadvantages of

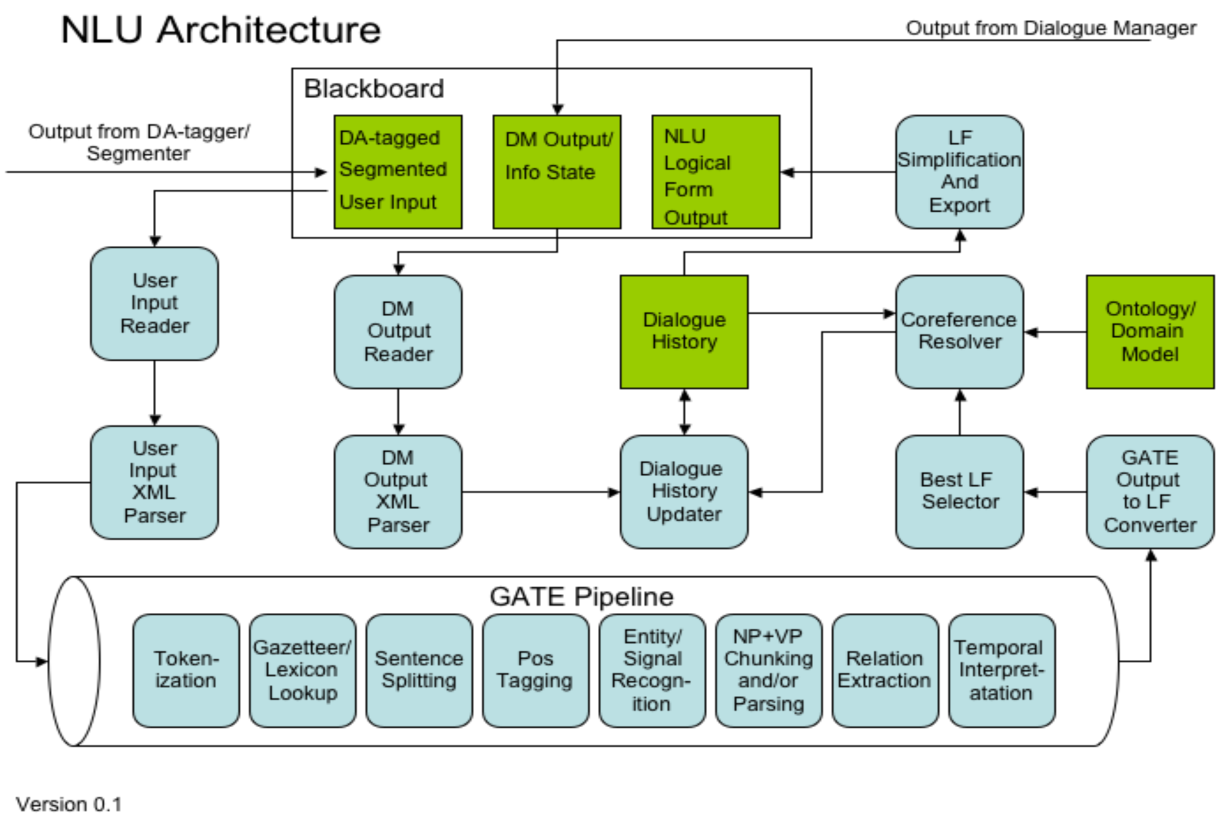


Figure 1 : Natural Language Understanding Architecture

each have been well-explored in the literature (McLeron et al, 2006) (Feldman et al, 2006) (Riloff 1996). Given the impossibility of obtaining significant amounts of annotated training data, we decided to explore a data-driven approach to manually creating rules for our IE system. This paper describes this approach, focusing on relation extraction, including the data sources we used and a special tool we created to support the creation and modification of rules for our system. The outcome is a methodology which supports the rapid development of a wide coverage entity and relation extractors for shallow dialogue understanding.

The existing research on using Information Extraction for Dialogue is more akin to finding database style information (Flycht-Eriksson, 2003) whereas our application to use Information Extraction to extract meaning from Dialogue chat that is not goal-directed is, we believe, novel.

2. Overview of the Methodology

We start by identifying all the Entities and Relations that are significant in our domain and those words or phrases that we believe trigger each Relation – called Relation Signals. We suppose entity extractors exist already – these being either well-known named entity types or other semantic classes that can be extracted using standard entity extraction techniques. For each relation, we proceed as follows:

1. Gather a pool of examples of utterances expressing the relation by searching the data sources (next section) using the trigger/signal words already identified. Negative examples are important here too (example containing the trigger word(s) but not expressing the intended relation.
2. Select an example from the pool and pre-process using a pipeline of GATE modules.
3. Input the example and associated annotations resulting from the preprocessing into the JAPE editor.
4. Manually create a JAPE pattern action rule in the editor by specifying (a) which of the annotations should form part of the pattern portion of the rule and (b) what output annotations should be added by the action part of the rule.
5. Run the new rule from 4 over the set of examples and refine the rule by further generalization/specialization to cover correctly as many of the positive examples as possible and as few of the negative ones as possible.
6. Remove the examples covered by the rule from the example pool and go to 2.

Note that this is a classic covering algorithm, well-known in the machine learning community (see, e.g. Mitchell (1997)). Here, however, we employ a human in the loop to carry out the generalization.

3. Dialogue Data on the Web

Once we have manually identified the basic set of Entities, Relations and Relation Signals, we can expand the coverage by finding more examples of dialogue turns (from a dialogue corpus) containing the relation signals for each of our relations. Several useful internet sites that contain dialogue corpora are :

- The Linguistic Data Consortium's corpus of Spoken Office Dialogues[1] :
- The Dialogue Diversity Corpus [2] **;**
- Saarbrucken Corpus of Spoken English [3].

We have also done some preliminary work using movie/TV scripts. We have used some of the scripts from the TV series Friends[4] which looks very promising for collecting non-goal driven dialogue turns.

For example, for the ATTITUDE relation discussed above, we found over 200 examples in two dialogue corpus sources.

Here is a sample from LDC corpus of Spoken Office Dialogue:

Um, th- what would - would - would - what would <Emphasis> ***worry*** *</Emphasis> me is that maybe we might miss a little detail*

we're not exactly sure either. So, don't ***worry*** *too much about it. <Comment Description="while laughing"/> The - It's just self rating.*

O_K. So, then in terms of people ***worrying*** *about,*
then we can start worrying about how we would

And <Emphasis> then </Emphasis> we can start ***worrying*** *about where to get this input, what -*

<Emphasis> that </Emphasis> I don't think is <Emphasis> even </Emphasis> worth us <Emphasis> ***worrying***

</Emphasis> about just yet. ***worry*** *about converting it to*

<Emphasis> English </Emphasis> and ***worry*** *about how it could ex- extract the parameters we need for the <Emphasis> belief-net. </Emphasis>*

uh, although we haven't ***worried*** *about this yet, you might wanna* ***worry*** *about something that would*

And here is a sample output from Friends scripts:

Friends15.txt:CHAN*: It doesn't matter. I just don't want to be one of those guys that's in his office until twelve o'clock at night* ***worrying*** *about the WENUS.*

Friends15.txt:RACH*: No. But don't* ***worry****, I'm sure they're still there.*

Friends21.txt:Rachel*: Oh God, oh. Great, Monica, y'know what, you could've called, I have been up here, I've been* ***worried****...*

Friends21.txt*:*Fake Monica*: Monica, I started my day by peeing in front of twenty-five other women, and you're* ***worried*** *about who's gonna take you to the Big Apple Circus?*

Friends21.txt:Monica*: Well, not...* ***worried****, just... wondering.*

Friends23.txt*:*RACH*: No, honey, they're not, but don't* ***worry****, because we are going to find them, and*
until we do, we are all here for you, ok?

Friends3.txt:MONICA*: No, he'll be fine. It's the other five I'm* ***worried*** *about.*

4. Creating Extraction Rules

Given such examples we then apply our method, as described above in section 2, for creating pattern action rules which match the examples and extract key relational content. Here we describe the creation of a single rule, driven by a set of seed examples.

4.1 Preprocessing/Feature Extraction

First the seed example(s) is passed to a pipeline of GATE (Cunningham et al, 1997) modules which performs linguistic preprocessing, but which may also be thought of as feature extraction. These modules include:

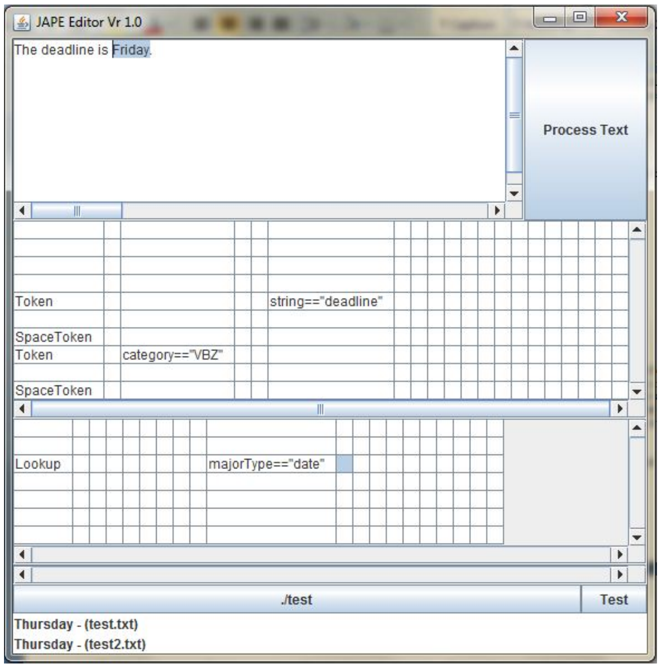
- Gazetteer/Lexicon lookup – matches token sequences against pre-stored lists of named entities (person, places, organizations) and lexical items in various semantic categories
- Sentence splitting – splits document into sentences
- POS tagging – assigns POS tags to each token in each sentence
- Entity/signal recognition – identifies and types token sequences as entities or relation signals in the domain of interest.

[1] www.ldc.upenn.edu/
[2] www-rcf.usc.edu/~billmann/diversity/DDivers-site.htm
[3] www.uni-saarland.de/fak4/norrick/scose.html
[4] ufwebsite.tripod.com/scripts/scripts.htm

- NP+VP Chunk Parser –identifies and types token sequences as NP or VP chunks and determines grammatical dependency relations between words
- Relation extraction – uses evidence from previous stages to determine whether there are instances of relations in the domain of interest
- Temporal interpretation – interprets information about temporal expressions from the entity recognition stage to assign calendrical time values, where possible, to temporal expressions

4.2 The Rule Editor

The extraction rules are written in the JAPE language – the Java Annotation Patterns Engine is a pattern-action rule language that supports creation of rules that match and add annotations to linguistically annotated data[5]. The rules are created and modified using the JAPE Editor, a tool especially developed for the current application.



The deadline is Friday.

Token string=="deadline"
SpaceToken
Token category=="VBZ"
SpaceToken

Lookup majorType=="date"

```
Thursday - (test.txt)
Thursday - (test2.txt)
```

Figure 2 : Jape Editor Graphical User Interface

The editor (as seen in Figure 2) is composed of three parts; the top part is the input box where the user is requested to enter a textual example. Apart from entering a sample sentence, he is also asked to highlight with the mouse which of the elements needs to be extracted by the algorithm. As soon as she is done and the “Process Text” button pressed, the sentence is sent to the GATE pipeline and passed through the various functions described earlier. The whole extraction rule process, including preprocessing, is actually controlled from within the JAPE Editor.

[5] See http://gate.ac.uk.

This brings us to the middle part of the rule construction grids - made up of three grids. The middle grid represents the analysis performed by the GATE pipeline on the word selected in the top window by the user. This might include the part-of-speech value of the selected text, semantic annotations (such as if the word is actually the name of a person), etc. The other two grids perform exactly the same function but they show only the analysis for the text before the selection and the text after the selection. So if we look at the example sentence “The deadline is Friday.” where Friday was selected, the first grid would show the processing for “The deadline is “, the middle grid would show the processing for “Friday” and the last grid would simply process the full stop found after the word Friday. If we just focus on the middle grid, we’ll find a variety of information such as the fact that Friday is a word, it starts with a capital letter, etc. This kind of information is so general that it is useless for building the Information Extraction pattern, so these features are simply discarded by the user. To discard such information, the user simply right clicks on the mouse and selects the delete option. The system offers various options in its menu, such as the facility to modify the existent values, add new conditions, etc. If we go back to pruning the output of the IE, we immediately notice that the semantic tag which links Friday to a date is a rather important clue, so that is retained as can be seen in the diagram (Fig 2). The user continues pruning and modifying the pattern until she is happy with the result. However, the effectiveness of such a rule can only be evaluated on a test corpus.

This is the role of the last box in the interface, which is the testing box whereby the rule is tested on a set of documents containing examples from the test corpus. Essentially, the user simply presses the Test button and the rule constructed through the interface is automatically applied to the document set in a specified directory. The result is shown in the adjacent box together with the name of the document from which it was extracted.

Once we obtain a simplified rule from the system, we then need to bootstrap with more examples. To achieve this, our dialogue corpora are used to find further examples. As mentioned previously, we are not limited to traditional dialogue corpora, but are also looking at film/tv examples found on the web. These are a rich source of conversational dialogues, albeit, they mimic real world dialogues within a particular genre : humor, drama, tragedy, etc. Nonetheless, they still contain a series of important examples which can be used. To identify/organise the newly collected examples, a concordance module is used to align the examples by the Relation Signal. With these new examples, we make use of the newly generated rule and we try to generalise over further examples. The idea is that from a base rule, we will be capable of covering more examples using simple modifications of that rule. This is something we learned from experience because when we handcrafted the rules, it was obvious that minor modifications of the base rules allowed the system to increase its coverage by several orders of magnitude. The most successful pattern is then identified and stored. This pattern will be used as a new JAPE rule inside our system. The examples which were identified using our concordance system, but which were not covered by the JAPE rule generated are extremely important and are used as new seeds in the subsequent iteration of the system.

5. Using the Paraphrase and Textual Entailment work done in IE

The method we have outlined has a serious limitation – it aims to expand the different ways of expressing a Relation given a Relation Signal, but does not identify other ways of expressing the same meaning with other Relation Signals. To approach this problem, we are studying the work done in IE on paraphrases – finding equivalent ways of expressing the same concept . Some approaches make use of identifying the same Named Entities in similar news items (Yusuke et al. 2002) however we cannot use exactly this approach since a dialogue is generally made up of a limited number of repeated Named Entities. Thus we cannot find similarities between sentences using Named Entities. Using the process described in section 4, we will increase our number of seed examples and use these results to discover new Relation Signals by applying work in finding paraphrases for IE. We are particularly interested in the work of discovering paraphrases using Machine Translation (Finch et al. 2005) to help discover different ways of saying the same thing : translating the target sentence/turn in language A into language B and then translating the sentence in language B back to language A. So using our example with Google Translator, if we translate "I'm a bit worried" into Czech and then translate the Czech, "Mám trochu strach." back to English, we get "I am a little scared." which does in fact given us another way of saying the same thing. Our plan is to try to use this approach for corpus examples containing our base Relation Signals to discover new Relation Signals. This work applied to our task looks promising and we hope to be able to report results on this in the near future.

The other work we are looking at is the work on textual entailment, which also addresses the issue of alternative ways of conveying the same meaning, perhaps through saying something which entails something else, i.e. where an entailment holds between two text variants that express the same target relation (Dagan et al. 2006)

6. Acknowledgements

This work was funded by the Companions project (www.companions-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-034434.

7. References

Cunningham, H., Humphreys, K., Gaizauskas, R., Wilks, Y. (1997). GATE -- a TIPSTER based General Architecture for Text Engineering. In Proceedings of the TIPSTER Text Program (Phase III) 6 Month Workshop. DARPA, Morgan Kaufmann, California.

Dagan, I. Glickman, O. and Magnini. B. (2006). The PASCAL Recognising Textual Entailment Challenge. In Quiñonero-Candela, J.; Dagan, I.; Magnini, B.; d'Alché-Buc, F. (Eds.) Machine Learning Challenges. Lecture Notes in Computer Science , Vol. 3944, pp. 177-190, Springer.

Feldman, R., Sanger, J. (2006) The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press.

Finch, A., Hwang, Y, S, and Sumita, E. (2005). Using machine translation evaluation techniques to determine sentence-level semantic equivalence, 3rd International Workshop on Paraphrasing (IWP2005), South Korea

Flycht-Eriksson, A. & Jönsson, A. (2003) Some Empirical Findings on Dialogue Management and Domain Ontologies in Dialogue System - Implications from an Evaluation of BirdQuest. *Proceedings of 4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan.

McLeron, B., Kushmerick N. (2006) Transductive Pattern Learning for Information Extraction. In Proc. of EACL 2006 Workshop on Adaptive Text Extraction and Mining, Trento.

Mitchell, T., (1997) Machine Learning, McGraw Hill Publishers.

Riloff, E. (1996) Automatically generating extraction patterns from untagged text, in Proc. AAAI-96, Portland, OR, 1996, pp. 1044–1049.

Yusuke, S., Sekine, S. (2002), Automatic paraphrase acquisition from news articles, Proceedings of the second international conference on Human Language Technology Research, San Diego, California, Pages: 313 – 318.