Workshop Proceedings of the 9th International Conference on Intelligent Environments
J.A. Botía and D. Charitos (Eds.)

doi:10.3233/978-1-61499-286-8-530

Beyond Artificial Intelligence Markup Language (AIML) Rapid prototyping of a Q&A system

Ioannis DOUMANIS[1] and Serengul SMITH
Middlesex University, The Burroughs Hendon, London NW4 4BT

Abstract. Although Q&A systems have attracted a lot of research interest and efforts during the past few years (e.g. [1] and [2]), there are still a limited number of freely available tools for rapid application prototyping. This paper reviews and compares the available resources and suggests the Virtual People Factory (VPF) [3] as a good solution to rapidly built such systems. It then suggests a novel natural language processing (NLP) algorithm, as an additional layer to increase the VPF accuracy. Finally, the paper discusses a short study where the two approaches (script-based vs. language parsing) are compared and draws conclusions on their performance.

Keywords. Natural language processing, question-answering systems, cultural heritage

Introduction

Preparing a Question & Answering (Q&A) system to accept free-form questions is a difficult and time-consuming process. As an example, we created a Q&A system in the domain of cultural heritage that accepts natural language questions about a medieval castle in Greece. The system was created to be capable of free-form Q&A about a limited number of locations in the castle with users in a lab environment [4]. The system's development took approximately 3 weeks and 100 hours of work to develop a conversational model with 59% accuracy. In this paper, we present the review we performed on a number of suitable tools for the development of the prototype. We selected these tools based on two requirements: a) free availability and b) rapid application development. Then, based on our comparison we conclude with Virtual People Factory (VPF)[3] as the tool to be used in the development of the prototype. The Virtual People Factory (VPF) is a freely available web application that offers an easy-to-use interface for the rapid development of Q&A systems.

Developers with minimal programming skills can create a conversational model online and integrate it with any desktop or web application with ease. However, as the VPF does not process language its limitations become apparent fairly quickly. Because the system does not recognize parts of speech (POS) for instance, it fails to distinguish keywords serving different syntactical functions in different user input. Further, the

[1] Corresponding Author: *Middlesex University, The Burroughs Hendon, London NW4 4BT;* Email: ioannis.doumanis@lsst.ac

absence of a dialogue manager makes it difficult to keep the topic of a conversation for several dialogue turns. To address these limitations we designed a natural language processing algorithm and a dialogue manager. We fully implemented the algorithm but not the dialogue manager. The algorithm adds three additional processing layers to the VPF web service. We detail our NLP approach and the benefits it adds to VPF. Finally, we present a short study where we compare our algorithm with the VPF service and draw conclusions on their performances.

1. PandoraBots

PandoraBots [5] is a free-to-use Artificial Intelligence (AI) web application used mainly to develop and publish software robots (or chatbots) on the World Wide Web. It has a large user community with more than 200,000 chatbots in multiple languages. Pandorabots is considered the representative of contemporary chatbot language processing. The service has been recently used in mobile applications to allow users to manage information through natural language commands. A very good example, is Jannie [6], a mobile virtual assistant that can answer natural language questions, send emails, play music automatically and much more.

The conversational engine of Pandorabots is based on AIML (Artificial Intelligence Modelling Language), an XML-based language created by Dr Richard Wallace [7] for creating conversational systems. The fundamental unit of knowledge in AIML is a pattern-template pair (see Figure 1), where patterns match the user's input and templates produce a response to the user's input matching the corresponding patterns. Pandorabot developers have the option to load a knowledgebase of 40,000 patterns in any newly created conversational system. However, any additional piece of knowledge needs to be written in AIML using an online editor. Further, AIML is not designed to process language and to understand the meaning and structure of words and sentences.

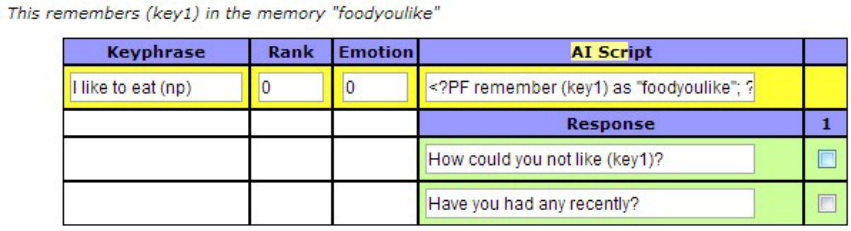
```
<category>
        <pattern>WHAT ARE YOU</pattern>
        <template>
                <think><set name="topic">Me</set></think>
                I am your intelligent tour guide of the Monemvasia castle
        </template>
    </category>
```

Figure 1. Example of an AIML category

2. The Personality Forge Engine

The Personality Forge Engine [8] is another yet more sophisticated free-to-use chatbot hosting service. It follows the same pattern-based approach as Pandorabox where the user's input is matched against predefined pairs of patterns in the database. However, as opposed to Pandorabox developers enter knowledge in the system in plain English using a simple web form. A custom-made scripting language is also available (called AIscript) for advanced developers to further extend the conversational capabilities of

the chatbots. In the sample code shown in Figure 2, developers can store the food the user likes in the bot's memory. Further, the system processes the user's input for its correct syntax before determining the best response. According to the official web site, the Personality Forge's AI Engine has a build in knowledge of over 150,000 words.


This remembers (key1) in the memory "foodyoulike"

Keyphrase	Rank	Emotion	AI Script	
I like to eat (np)	0	0	<?PF remember (key1) as "foodyoulike"; ?	
			Response	1
			How could you not like (key1)?	
			Have you had any recently?	

Figure 2. An AIScript example

3. The Virtual People Factory(VPF)

The Virtual People Factory [3] is a web application designed to educate Pharmacy/Medical students in effective patient-doctor communication skills. The system allows the development of conversational models that simulate various medical scenarios (e.g., a complete neurological examination on a virtual patient). The usefulness of VPF is not limited to medical scenarios. It can be used to simulate any scenario where dialogue is needed. Similar to the other two systems, VPF follows a pattern-based approach to match the user's input to the database. However, it differs from Pandorabox and Personality Forge in a number of ways:

1. Both Personality Forge and PandoraBots require rules to be said verbatim to match the input string. On the other hand, VPF uses a matching heuristic to determine the similarity of the input (OK, I am ready lets begin the tour) to an entry in the script (lets begin the tour).
2. Because of the above approach in input matching, VPF requires fewer rules to answer the same output in comparison to both Personality Forge and PandoraBots. This has a significant impact on the systems performance and management of scripts for large application domains.
3. VPF, in contrast to Personality Forge and PandoraBots, enables a developer to define how-well a rule should match the input and cut-off any matched-rules below that threshold level.
4. Creating a good script in PandoraBots and Personality Forge requires extensive knowledge of each engines internal scripting language. VPF scripts, on the other hand, are written in plain English.
5. VPF considers the information space in terms of acts (very large chunks of information) and topics (smaller chunks of information). PandoraBots divides the information space only into topics. Personality Forge does not create subsets of input data as topics of conversation readily.
6. VPF provides an easy-to-use system to deal with the systems failed responses. The absence of a similar system is perhaps the biggest weakness of Personality Forge as it makes correcting failed output from the system a very difficult task.

7. VPF endows developers with full control over a response of the system. However in Personality Forge, the AI engine takes control of the systems output with random responses quite often being produced.
8. The VPF in contrast to Personality Forge offers a reliable web environment for development and testing of Virtual Humans. The low data transfer speeds of Personality Forge limit the usefulness of that service.
9. Contrary to Personality Forge, VPF offers a free and easy-to-use API (Application Programmable Interface) for integration into applications. There is no limitation on the number of messages a program can exchange with the VPF server. On the other hand, the free version of ”Personality Forge” API is limited to 500 responses [9].
10. Although Personality Forge use world list wildcard rules, these are not associated with a single word and therefore are not automatically reusable throughout a script. VPF offers a simple but very intuitive Synonym-List finder that automatically associates the chosen keywords with synonym lists for the entire script.

Based on this discussion, it is clear that VPF is the tool of choice for our development needs. However, as discussed in the next section there is room for additional improvements to increase its accuracy.

4. NLP Algorithm

The workflow of the current implementation of the algorithm can be seen in Figure 3. It uses a four-layered approach to map an input string from the user to an appropriate response in the database. At the first-layer of processing is Virtual People Factory [3]. The designer defines the knowledge contained in a domain by entering (in plain English) pairs of questions & answers in the VPFs database. It provides the first processing layer for the input by computing its similarity to a question (called a trigger) in the database. Once the trigger is found, the system responds with the answer (called a speech) for the trigger [2]. A major step in the matching process is the use of a list of global keywords. These are the most important words used by triggers globally and are extracted in real-time when the designer enters the question & answer pairs in the database. However, as the keywords are not annotated with part of speech (POS) information, VPF fails to distinguish ambiguities between triggers that contain the same global keywords. For example, consider the following two triggers: a) shall we begin the tour? b) Can we tour now?

The third layer performs a deep syntactic analysis on the input string. It parses the input for predicates and deep syntax dependences using the Sanford Parser [12] and Antelope API [10]. It then searches for the best match of the parsed input against phrases in the database. The matching process involves several comparisons/matching tests with incrementally relaxed conditions. This ensures that if at least some of the predicate arguments of the input and the database are the same (or similar), matching will be successful. Once a match is found, the phrase is passed to the VPF for an exact match. If this step fails, the system either does not have a response, or it did not understand the question the way the user asked it. Hence, it replies with a generic off-topic response (e.g., I do not understand please rephrase or move to a different topic).

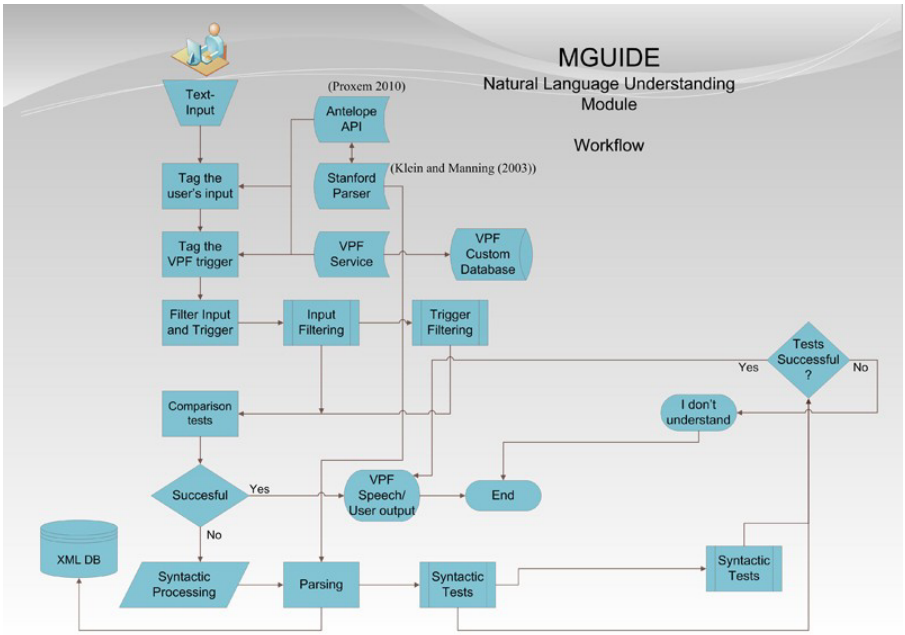


Figure 3. The workflow of the algorithm

We have also designed and partially implemented a fifth processing layer that was not included in the current algorithm implementation. It performs shallow semantic analysis of the input text. A shallow form of semantic representation is a case-form analysis, which identifies the sentences predicate (e.g., a verb) and its thematic roles (e.g., AGENT, EXPERIENCER, etc.). In a few words, this process assigns who did what to whom, when, where, why, and how to the input sentence. Currently, the modules semantic component is not mature enough to be used in a real dialogue application. It uses an open-source semantic parser [10] which is highly experimental. However, even if the parser is improved in future versions, it is unlikely that it will become powerful enough to resolve accurately the natural language ambiguities even in limited domains. Consider for instance, the utterance I want more information about the church. The subject I can be considered either as an AGENT (i.e., who performs the action) or the EXPERIENCER (who receives the result of the action) of the predicate, so there are two distinct case-frames. An experimental semantic processing stage in the current algorithm (discussed above) has been developed to address this problem (within limits). In particular, it uses a predefined library of valid case-frames in the domain of the prototypes (e.g., frame want, frame see, etc.) in order to automatically cut any invalid interpretations. With this constraint, it then searches for specific thematic roles (and their values) to help make sense what is being discussed. For example, the previous sentence maps to frame want with thematic roles and values: PATIENT: Information, GOAL: Church. Once the same case name and a case component with the same label and value match, the utterance for that frame is returned. Of course, more research and development is needed to refine this stage, but it can currently match correctly a range of questions (and paraphrases) to the corresponding frames in the sample database. The code for this stage will be released as open source, along with the rest of the algorithm.

5. Dialogue Manager

The dialogue manager is modelled as a Hierarchical Tasks Decomposition process: acts, topics, subtopics, and their associative trigger templates. Trigger templates are framed-like structures with slots representing a trigger phrases case frame, predicate argument structures and POS keywords. For each trigger phrase in the database, a separate template is defined.

This process is being carried out semi-automatically, where the system generates the templates automatically and the designer manually corrects any failed or multiple interpretations of the trigger phrase. This hierarchy allows the system to keep the context as it has detailed information on what has been activated at each level of the conversation (e.g. POS keyword(s)). Figure 4, shows a generated graph for the domain of the prototypes (i.e., a tour of a medieval castle) along with two trigger templates for the phrase "I want more information about the central gate".

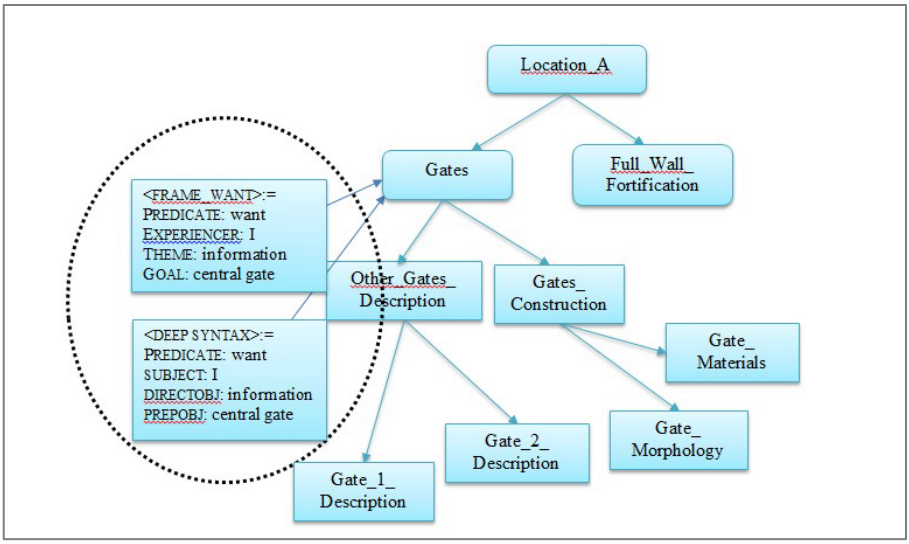


Figure 4. A sample graph generated by the dialogue manager

Each of the above nodes carries an activation list, where the developer specifies:

1. How many times a node should be activated. For example, the Introduction nodes are activated only once. This way, the ECA can understand when the greeting time is over and the real conversation begins.
2. Prioritize the activation of the nodes using a priority value. For example in the graph of Figure 4, the node Gate Morphology logically has a greater probability to be next in the discussion than the Gate Materials. This is because a user will most likely ask questions about the form of the gate first, before getting into questions about the materials that were used in its construction. This prioritization value for each node is difficult to determine, and should be empirically determined through testing.

3. Define each node activation preconditions. For example, the Gates Construction node can only be activated if the node Gates has been activated first.
4. How the system responds when the above conditions are not met. For example if the greeting time is over and the user says hello, a reply could be "Ioannis, how many times are you going to say hello to me".

6. Algorithmic performance

The current implementation of the algorithm has two layers (a shallow parsing and deep syntactic processing layer), to map the users input to a proper response in the database. Although parsing is more precise than the script-based approach, it needs much processing power. As it is not always possible for mobile devices to have a stable Internet connection for processing to take place in the cloud, some of the processing should be conducted locally. For this reason, we sought to compare scripts with shallow parsing (scripts vs. shallow parsing) to get some insights into the robustness of each method. Furthermore, we compared each of these methods with the deep syntactic processing approach in order to investigate further what is lost when precision is sacrificed for robustness. We had the following hypothesis:

H1: The deep syntactic processing approach does not overall outperforms a script-based approach, but it is better for processing more complex questions.

H2: There is no overall performance difference between the script and the shallow parsing approach. However, an overall performance difference between the shallow parsing, and the deep syntactic processing approach is expected.

Using the end-user logs from a pilot experiment we conducted using the prototype Q&A system [4], we extracted 60 questions which the systems failed to answer and asked an expert to create their responses. These new sets of stimuli-responses were used to augment the existing corpus using the VPF tool. There is evidence in literature [11] that the use of both end-users and domain-specific experts in the process of conversational modelling provides a more comprehensive coverage of the conversational space than when the model is crafted by a developer alone. Therefore, the conversational corpus used by both systems should be sufficient enough to enable a more effective comparison of the methods used for processing natural language questions. The following methods were investigated:

- Scripts vs. shallow parsing
- Scripts vs. deep syntactic processing
- Shallow parsing vs. deep syntactic processing

In all conditions, a single user asked each system 60 random questions that covered the four locations in the castle for which the system was providing information and marked each system response using the following scale:

- Each correct answer received 20 points
- Each relevant answer 10 points

- Each irrelevant answer 5 points
- No points were given when the system returned a random answer (or no answer at all).

The total score (expressed as the percentage of the points given for achieving a perfect score) achieved gave the overall performance of each method. Table 1 shows the results:

Table 1. Algorithmic performance between the conditions

Comparison	Performance	
Scripts vs. shallow parsing	59%	57%
Scripts vs. deep syntactic processing	59%	57%
Shallow parsing vs. deep syntactic processing	57%	40%

There was a variation of performance for the deep syntactic processing method, across the content presented about the locations of the two routes (40% vs. 25%) (see Table 1 and Table 2). This effect was not observed in any other condition. This is clearly due to the unknown predicates used in the questions asked in the attractions of the second route. The predicate matching heuristic of the deep syntactic processing layer fails if the database does not contain the relevant predicates.

Table 2. Algorithmic comparisons per type and part of the tour

	Script Approach	Shallow Parsing	Deep Syntactic Processing
First part of the Tour	58%	56%	40%
Second part of the Tour	59%	57%	25%

In terms of overall performance, the results validate my original hypotheses (see H1 and H2). Although scripts questions with poor language skills, they are clearly more robust in providing overall better answers than the parsing approaches (i.e., both shallow parsing and deep syntactic processing). The slight difference in performance between scripts and the shallow parsing approach shows that the method can be used for filtering-out input-stimuli pairs that do not match grammatically and, therefore, provide more accurate answers. Furthermore, the results show that the deep syntactic processing layer performed below average. As the syntactic parser [10] is still not mature enough, it gave several failed parses of questions that dropped the overall performance of this layer. Therefore, it is reasonable to assume that once the performance of the parser is improved in future versions, the performance of this layer will be improved as well.

7. The Q&A System

We used the parsing algorithm to implement a simplistic Q&A system. The system provided participants with cultural content covering popular attractions on two routes in a real archaeological attraction and allowed them to ask questions using plain English after each presentation was complete. Each route included three locations to visit in turn. An expert human-guide wrote the presentations and crafted the initial conversation corpus using the Virtual People Factory authoring tool.

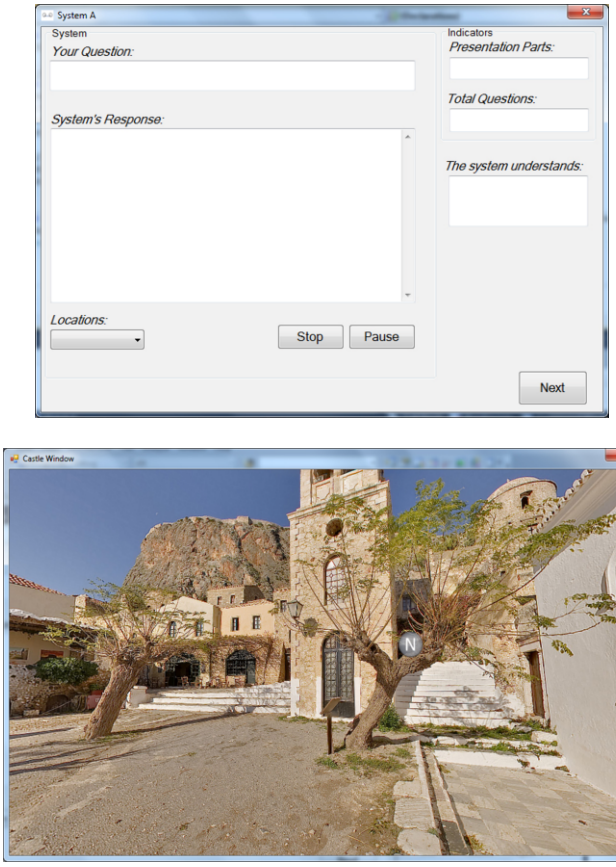


Figure 5. The prototype Q&A system

The interface of the system is simple enough to use without any previous training and it is divided into the following sections:

- **The system section:** This section features an input field for typing a question, an output field for displaying the system's response, a drop-down menu for defining the location the user is visiting, and two buttons for controlling the speech output of the system.

- **The indicators section:** This section provides information about the total number of questions asked, the database question the system matched the input question to, and the part of the presentation where the user is currently listening.

A simple key combination activates the display of an interactive panoramic representation of each location participants had to visit. The system was compared with a similar system featuring only the script-based algorithm in an empirical study in the lab. The results of this experiment are discussed in [4].

8. Conclusion and Future Work

This paper presented a more robust and linguistically motivated different option for the development of Q&A systems to the open-source tools currently available to the ECA research community. Developing and releasing the full algorithm and dialogue manager is a long-term goal. However, the current implementation of the algorithm will be released as open-source immediately for the benefits of the virtual human research community. Our current work involves developing an editor and a simple API, to allow developers to integrate with ease, Question & Answering (Q&A) functionality to their applications. The editor allows the designer to map sample questions-to-answers in a simple and straightforward way. It updates its internal databases automatically, while the user enters the question-answer pairs in the editor. The lexical information (e.g., predicate synonyms) required by the algorithm, are provided by the designer in the settings panel of the editor. Other more advanced features, like the threshold applied in each script, are accessible via the web interface of the VPF system. The API is compatible with all the recent windows operating systems (OS); integrated development environments (IDEs), which we hope will encourage the wider dissemination of the algorithm in developing systems. We welcome collaborations on the further development of the editor and the algorithm.

References

[1] Leuski, A., et al.: Building effective question answering characters. In: Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue (2006).
[2] Dickerson, R., Johnsen, K., Raij, A., Lok, B., Hernandez, J., Stevens, A., and D. Lind. (2005): "Evaluating a Script-Based Approach to Simulating Patient-Doctor Interaction" SCS 2005, International Conference on Human-Computer Interface Advances for Mod- elling and Simulating (SIMCHI 05), 23-27 January 2005, New Orleans, pp.79-84.
[3] Virtual People Factory (VPF) (2012) (Available from: http://www.virtualpeoplefactory.com/ VirtualPeopleFactory/) [Accessed December 31 2012].
[4] Doumanis, I. (2013): "Evaluatiing Humanoid Embodied Conversational Agents in Mobile Guide Applications". Thesis (PhD). Middlesex University
[5] PandoraBots (2012) (Available from: http://www.pandorabots.com/botmaster/en/home) [Accessed December 31 2012].
[6] Pannous GmbH, Jeannie, Android Mobile Application and manual, Hasloh, Germany, Europe.
[7] Wallace, R. (2003). The Elements of AIML Style. [Online]. Available from: http://www.alicebot.org/ style.pdf. [Accessed: 6/5/2010].
[8] Personality Forge (2012) (Available from: http://www.personalityforge.com/) [Accessed December 31 2012].

[9] Personality Forge Chatbot API (2012) (Available from: http://www.personalityforge.com/chatbotapi.php) [Accessed December 31 2012].
[10] Proxem Advanced Natural Language Object oriented Processing Environment., version 0.8.7 computer program and manual, Proxem, France.
[11] Rossen, B., Lind, S., Lok, B. (2009): Human-Centered Distributed Conversational Mod- eling: Efficient Modeling of Robust Virtual Human Conversations. In: Ruttkay, Z., Kipp, M., Nijholt, A., Vilhjlmsson, H.H. (eds.) IVA 2009. LNCS, vol. 5773, pp. 474481. Springer, Heidelberg (2009).
[12] Klein, D. and Manning, C.D. (2003): Accurate unlexilised parsing. In ACL '03 Proceedings of the 41st Annual Meeting on Association for Computational Linguistics – Volume 1, pp. 423 – 430.