Demo: Natural Language Processing for Online Fraud Scenario Extraction

Bas Testerink, Floris Bex[1]

Abstract. In this demo we present our progress in the project Intelligence Amplification for Cybercrime (IAC), in which we apply AI techniques to allow natural dialogues for online criminal complaints concerning fraud cases. We show the natural language processing and dialogue modules of the system. The dialogue module allows for mixed-initiative dialogues between human complainants and software agents for crime intake. An interface is provided that allows the complainant to input free text and form elements, which are then integrated into a structured knowledge graph by the NLP module. This knowledge graph then serves as input for the intake agent, who can use it to reason about the incident that has occurred and formulate follow up questions to the user. A paper with full details regarding the background of this demo application has been accepted in the AI4J workshop at ECAI 2016 [2].

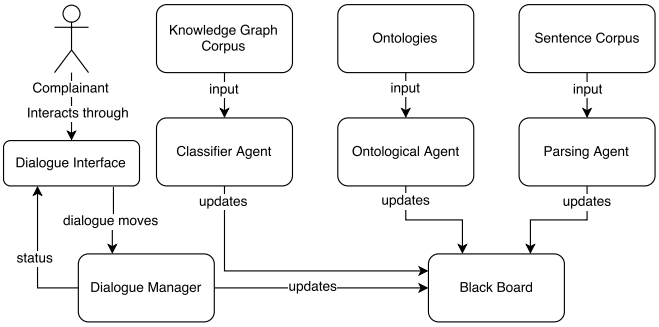

Figure 1. Architecture of the intake system. Boxes indicate software modules. Arrows indicate interaction between components such as service calls or input provision.

1 Introduction

Reasoning in police investigations is a complex process, which consists of collecting, organizing and assessing a mass of unstructured and unreliable evidence and scenarios in a case. Artificial Intelligence has proposed various scientifically founded ways of treating evidence using, for example, Bayesian networks or non-monotonic logics (cf. [6]). One of the elements that is missing from AI models of evidential reasoning is a rendering of the *process* of constructing and analysing scenarios, which can be seen as a dialogue between multiple agents (e.g. police analysts, witnesses, forensic experts). Another problem for logical or probabilistic models of evidence is that the focus in real cases is often on more linguistically oriented concepts such as *arguments* and *stories*, often rendered unstructured and informal, as natural language. What is hence needed are technologies and theories for the process of investigation that bridge the gap between dialogical natural language interfaces and more formal models of evidential reasoning.

The project Intelligence Amplification for Cybercrime (IAC) aims to develop technologies to bridge the gap between natural arguments and stories in dialogue and structured scenarios for evidential reasoning. This development has as a practical goal to improve the online intake of criminal complaints and the subsequent investigations on the topics of e-crime and cybercrime for the Dutch National Police.

Our demo shows the initial developments of our mixed-initiative, agent-based intake system. In particular, we demo a dialogue interface and a natural language processing module that translates structured and unstructured free-text input from the dialogue interface to a knowledge graph, a labelled graph containing the entities, events and relations in a case. We then show how the system can reason with these knowledge graphs to formulate further questions that can be inserted into the dialogue.

The demo is intended for those who are interested in dialogues, natural language processing and agent programming. The latter target group might be interested because the underlying implementation is created with the agent paradigm and implemented with the agent programming framework from Dastani and Testerink [4].

2 Application Description

An overview of the system is shown in Figure 1. The complainant and police interact with the system through a dialogue interface (Fig. 2). This interface allows users to submit input, i.e. make dialogue moves, but also shows the status of the dialogue such as the open questions.

Questions can be generated by both the complainant and the police, but will also originate from the system itself through the scenario reasoning module. The dialogue is managed by a dialogue manager that maintains the legal moves of the participants. The legality of a move for a participant is based on the participants' commitments in the dialogue (e.g. statements that were made). The maintenance of the commitments in a commitment store is also part of the dialogue manager and its details are explained in [1]. The natural language processing system is called upon in case a participant provides free-text input. This system also maintains a knowledge graph on a blackboard that is constructed throughout the dialogue and by different agents (or a single agent that contains all the modules). The graph serves as input for the scenario reasoning module of the application which then, based on the status of what is known about the reporter's incident, asks extra questions and clarifications through the dialogue manager. Finally, the scenario reasoning module also provides the analysis of reports and cases to the police.

[1] Utrecht University, the Netherlands, email: {B.J.G.Testerink, F.J.Bex}@uu.nl

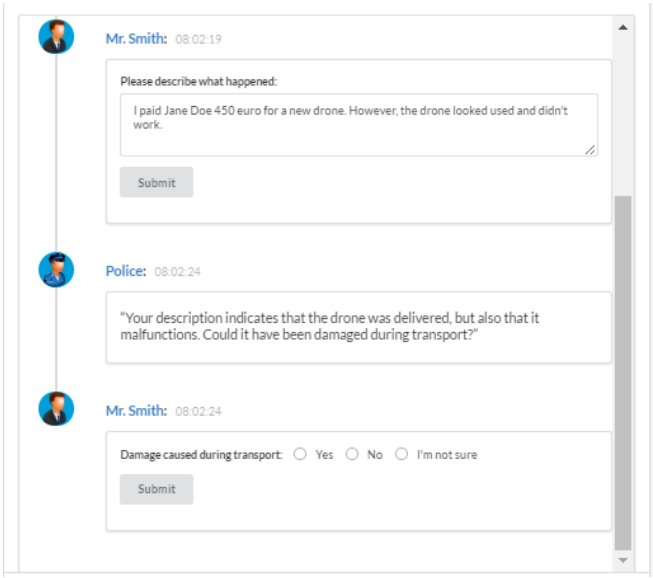

Figure 2. Screenshot of the dialogue interface

Our demo shows a web application in which a user can file a report and see how the knowledge graph is constructed. An example knowledge graph that is generated by the application is shown in Figure 3. In this example there is a submitted form that contained data such as addresses and names. This data is adopted as light blue entities and edges in the graph. The entities and edges are subject to an ontology - for clarity, things like class membership are not shown here. The form contains entities such as the complainant and the counter party. Also part of the form is the story that describes the incident. In this case we use a Dutch two sentence story: *"Ik heb Floris betaald. Maar hij stuurde het product niet op."* (translated: *"I had paid Floris. But he did not send the product."*).

This story is processed by the Alpino dependency parser [3] and its output is added to the knowledge graph as green nodes (for words and part-of-speech tags) and edges (dependency relations and head relations). Then, we apply a named entity recognition module which tries to find mentions of known entities from the form in the text (such as the counter party and the complainant), or identify new entities. Its

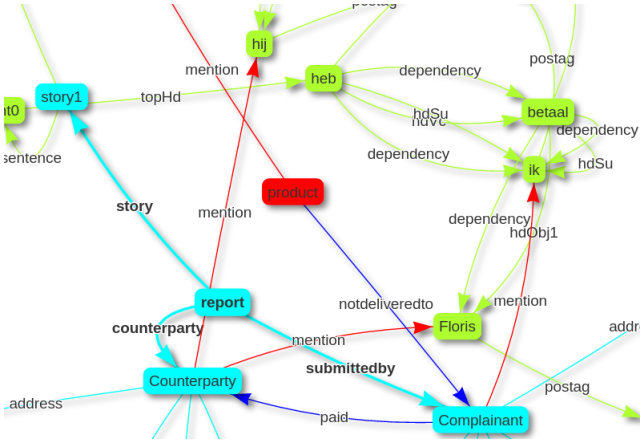

Figure 3. Part of a knowledge graph as outputted by the application.

output are red nodes and edges. In this example the module recognized the complainant in the text as being mentioned by 'Ik' (translated: 'I') and the counter party as being mentioned by 'hij' (translated: 'he') and his first name 'Floris'. Also a new entity 'product' was found in the text. The named entity recognition module uses the earlier adopted dependency tree information to find mentions. One can hand craft classifiers using SPARQL queries (which is part of the demo) or, as we intend to do at a later stage, apply machine learning techniques such as subgraph feature extraction [5] to train classifiers. A classifier may find certain paths to indicate certain relations. For instance, if a word 'I' is part of a story, which belongs to a report R, then there is an increased probability that 'I' is a mention of the complainant of report R, as complainants usually write in the first person. Aside from named entity recognition we can also apply other classifiers that try to find different relations among the identified entities. In the example these are adopted as blue edges and the classifiers here found that the complainant paid the counter party, and that the product entity form the text was not delivered to the complainant.

Once a knowledge graph has been built, we can perform reasoning over the graph and give feedback to the user via the dialogue interface. Given a number of templates or schemes for standard fraud scenarios (which will be implemented in the demo), we can test whether all the elements of the scenario have been correctly mentioned in the complainant's story. If, for example, the user does not mention a product in his story, then the system can ask the user to further specify this through the dialogue interface.

The different information sources are all implemented as agent modules that can be instantiated as individual agents or be combined in a large single agent. Alongside the knowledge graph we will show the structure of the source of the agents and discuss the benefits of the agent programming paradigm when one constructs a dialogue system such as ours. These include among others the ease of introducing new information sources to the graph and parallel computing.

References

[1] Floris Bex, John Lawrence, and Chris Reed, 'Generalising argument dialogue with the dialogue game execution platform', 141–152, (2014).
[2] Floris Bex, Joeri Peters, and Bas Testerink, 'A.i for online criminal complaints: from natural dialogues to structured scenarios', in *Proceedings of the workshop AI for Justice (AI4J) at ECAI 2016 (t.b.p)*, (2016).
[3] Gosse Bouma, Gertjan Van Noord, and Robert Malouf, 'Alpino: Wide-coverage computational analysis of dutch', *Language and Computers*, **37**(1), 45–59, (2001).
[4] Mehdi Dastani and Bas Testerink, 'From multi-agent programming to object oriented design patterns', in *Engineering Multi-Agent Systems*, 204–226, Springer International Publishing, (2014).
[5] Matt Gardner and Tom Mitchell, 'Efficient and expressive knowledge base completion using subgraph feature extraction', *Proceedings of EMNLP. Association for Computational Linguistics*, **3**, (2015).
[6] B. Verheij, F.J. Bex, S.T. Timmer, C.S. Vlek, J.-J. Meyer, S. Renooij, and H. Prakken, 'Arguments, scenarios and probabilities: connections between three normative frameworks for evidential reasoning', *Law, Probability & Risk*, **15**, 35–70, (2016).