Argumentation-driven information extraction for online crime reports

Marijn Schraagen
M.P.Schraagen@uu.nl
Information and Computing Sciences, Utrecht University

Bas Testerink
Bas.Testerink@politie.nl
Dutch National Police

Daphne Odekerken
D.Odekerken@uu.nl
Information and Computing Sciences, Utrecht University

Floris Bex
F.J.Bex@uu.nl
Information and Computing Sciences, Utrecht University

ABSTRACT

A new system is currently being developed to assist the Dutch National Police in the assessment of crime reports submitted by civilians. This system uses Natural Language Processing techniques to extract observations from text. These observations are used in a formal reasoning system to construct arguments supporting conclusions based on the extracted observations, and possibly ask the complainant who files the report extra questions during the intake process. The aim is to develop a dynamic question-asking system which automatically learns effective and user-friendly strategies. The proposed approach is planned to be integrated in the daily workflow at the Dutch National Police, in order to provide increased efficiency and transparency for processing of crime reports.

KEYWORDS

Argumentation, Information Extraction, Relation Extraction

ACM Reference Format:
Marijn Schraagen, Bas Testerink, Daphne Odekerken, and Floris Bex. 2018. Argumentation-driven information extraction for online crime reports. In *Proceedings of International Workshop on Legal Data Analytics and Mining (LeDAM 2018).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The ideas presented in this paper are part of a a collaborative initiative of the Dutch National Police and Utrecht University for developing a framework for (semi-)autonomous business processes in the police organization using technologies from text and data analytics together with computational argumentation and dialog. One project under the umbrella of this initiative concerns technologies to improve the intake of criminal reports submitted by civilians on the topic of online trade fraud, which concerns cases such as fake webshops and malicious second-hand traders on trading platforms (e.g., eBay). Around 40.000 reports are filed each year, and the legal background for trade fraud is a single article of the Dutch Criminal Code (art. 326) and a relatively small set of cases that are used as legal precedents. This high volume and relative simplicity of such cases makes them ideal for further automated processing.

For the case of online trade fraud, the Dutch police currently collects online-submitted crime reports using a web interface which requires citizens to fill out several predefined fields (such as the name of the counterparty, bank account number, etc.) as well as a free text description of the situation. Using this information the police decides to either (a) discard the report because it does not concern trade fraud, (b) accept the report and include it in the police database for further processing, or (c) ask follow-up questions (by e-mail) to the complainant in case more information is needed. In the current situation, human analysts have to read through all incoming reports and decide on either (a), (b) or (c). To improve the efficiency of this assessment, we aim to develop a system that automatically determines the appropriate course of action given a report.

One way of handling (possible) trade fraud reports is to train an algorithm to automatically determine which action to take given a complete incoming report. This was explored in previous research [4, 5], where classifiers were trained to classify reports as being of class (a - discard report) or of class (b - accept report), based on the elements of the report (address of suspect, trade site that was used, shallow linguistic features). Given that the data is highly skewed – only 16% of the incoming reports is normally discarded by human analysts – the results are promising, with an F_1-score of 67.5% for class *discard*, 95.2% for class *accept* and a macro-average F_1-score of 80.8%.

One important issue with the above solution is that for a machine learning classifier it cannot be explained satisfactorily *why* a complaint was discarded or accepted. For example, one important feature that is used as input for the final classifier FC algorithm is the output of another classifier WC trained on the (lemmatized) words of the free text field. The explanation of FC's decision to accept a report is then, for instance, that the classifier WC gives a probability of 0.8 to accept, based on the occurrence of certain words (such as “never” and “tickets”) in the report text. In a legal or law enforcement application, however, we need transparent explanations that make sense from a legal and common-sense perspective, not explanations that are based on certain patterns in the data. For example, we want to know that the complainant who filed the report bought tickets from the (suspect) counterparty, but these tickets were never delivered.

LeDAM 2018, October 2018, Turin, Italy

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

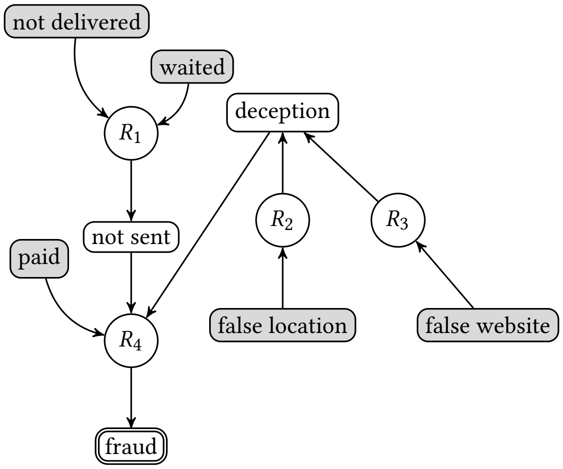

Figure 1: Example argumentation graph.

In order to automatically assess trade fraud reports submitted online, we turn to a combination of symbolic, argument-based reasoning about a case (similar to [7]) and non-symbolic information extraction techniques that use machine learning. These extraction techniques are intended to find basic observations such as “this report concerns a ticket for a music concert”, “money was paid by the complainant to the counterparty” and “nothing was delivered to the complainant”, and use these observations as premises in legal arguments to infer that, for example, the report concerns a possible case of fraud and should therefore be considered for further processing. Thus, the non-symbolic algorithms are fine-grained: the basic observations are closer to sentences in the original report texts, so their occurrence can be explained by exactly those sentences, and more complex conclusions based on multiple factors in a case can be checked by means of the argumentation.

In the rest of this paper, we discuss the concepts of our intake system. The design and implementation of the system is part of ongoing research, therefore the discussion in the current paper is primarily intended to be conceptual (leaving a full evaluation for future work). The current discussion is structured as follows: Section 2 discusses the argumentation theory and inference mechanisms that constitute the basis of the automated reasoning about fraud. The process of collecting complaint information can be modeled in different ways, which is described in Section 3. One of the proposed approaches involves a dialog with the complainant, which requires a question asking policy, as described in Section 4. As a prerequisite for argumentative inference, the basic observations need to be extracted from the input given by the complainant. For textual input, natural language processing (NLP) techniques are required for this task. The observations as used in the graph generally denote a relation between entities, e.g., a *send*-relation involving the *complainant*, the *counterparty* and a *package* as relation elements. The classification of entities and relations is described in Section 5. Section 7 concludes the paper and discusses next steps.

2 ARGUMENTATION THEORY

The Dutch Criminal Code defines fraud as “misleading through false contact details, deceptive tricks or an accumulation of lies”. These elements can be traced back to observations or observable facts collected from the victim and relevant third parties. Based on the legal definitions in the Dutch Criminal Code, the relevant case law and knowledge of working procedures of the police analysts who currently assess the fraud reports, we have constructed an argumentation theory about online trade fraud. To construct the argumentation theory, the right balance needs to be found in the level of detail for observations. On the one hand we want an observation to be directly observable from the input document, for instance ‘no mention of payments occurs in this document’. On the other hand, observations that are too detailed lead to a large argumentation theory, which is more difficult to construct, maintain and use in argument inference. We try to find a balance by interacting with the police-side users of the system such as the people that handle incoming complaints. If they think a statement is obvious from a document then we do not require an argumentation structure for those statements. Such statements are candidates for becoming observables. For other statements such as ‘this document concerns fraud’ it is not immediately obvious and we require some argumentation as to why a crime is committed in that case. Currently, we work with an argumentation theory of 46 rules and 26 observable facts [1].

The argumentation rules and observables can be modeled in an argumentation graph, where (sets of) observations provide support or counter-evidence for other propositions. A simplified example argumentation graph is presented in Figure 1. Inference rules are conjunctive, e.g., ‘*if* the package is not delivered *and* the complainant waited a for reasonable period of time *then* the package is not sent’ (R_1). The observation nodes are indicated with a gray background in Figure 1.

Once the graph is constructed, the observed nodes are used as input to infer conclusions. As per ASPIC$^+$ [6] definitions, we use inference trees as the data structure with which to represent arguments. An inference tree consists of a set of premises and a conclusion connected by rules, with possible intermediate conclusions (which are in turn premises for further conclusions) in between. For example, in Figure 1, the inference tree for the conclusion *not sent* contains the premises and the conclusion of rule R_1, whereas the inference tree for the conclusion *fraud* contains all nodes in the graph. Arguments may attack each other because of inconsistent conclusions (rebutting attack) or because a conclusion contradicts a premise of another argument (premise attack). Given a set of arguments and the attack relation, we determine the set of acceptable arguments by calculating the grounded extension from Dung’s abstract argumentation framework [3]. The grounded extension contains all arguments that are conflict-free and that defend themselves against any attackers, that is, if argument A in the grounded extension is attacked by argument B which is not in the grounded extension, then there is an argument C in the grounded extension that attacks B and thus defends A. Other options than grounded semantics exist, but grounded semantics fit nicely with the conservative nature of legal processes and can be computed in polynomial time given the arguments.

Consider for example the situation that a package has been sent, however the recipient was not at home and the delivery service issued a note that the package has been returned to the sender. For the purposes of the example, assume that the counterparty in fact has bad intentions (e.g., sending a defective product) and has used a false address. In this case the propositions *false location, not delivered, waited* and *paid* are true, but *not sent* is observed to be false (given the note from the delivery service). Based on these observations and the argumentation graph presented in Figure 1, using a forward chaining algorithm the conclusions *deception*, *not sent* and *fraud* can be inferred. However, the conclusion *not sent* conflicts with the observation *sent*. Therefore *not sent* and the dependent conclusion *fraud* are not in the grounded extension, which consists of the observation set and the conclusion *deception*.

When sufficient information is available the argument inference will result in a *stable state*[1]. We say that a certain conclusion is stable if either A) an argument for it is included in the grounded extension and more information does not change this, or B) there is an argument for the conclusion but this argument or any other argument for the conclusion can never be in the grounded extension, or C) no argument can be made and neither will this be possible with more information. For instance, consider a case where the counterparty in the case has refunded the payment to the complainant. In that case, there is no legal basis anymore to convict the counterparty of fraud. So if this proposition is observed for a case, then the system can establish that there will never be an argument for fraud in the grounded extension. The information necessary to result in a stable state needs to be provided by the complainant (possibly combined with information from third parties, such as banks or trade websites). The interaction with the complainant can be modeled in different ways, which will be discussed in the next section.

3 USER INTERACTION

As mentioned earlier, the Dutch police currently collects a report (including free text but also predefined fields for addresses, trade sites, etc.) using a web interface. The argumentation system as described in Section 2 can be based on this document by providing a conclusion (i.e., *fraud* or *not fraud*) if a stable state is reached, and suggesting to ask follow-up questions otherwise. Here, the full report document is used as input to instantiate propositions in the argumentation graph.

Alternatively, the user interaction model can be changed into a dialog paradigm. In this case the complainant does not file a report document, but instead the system guides the complainant through the reporting process by asking a number of questions. After each question the argumentation graph is updated using the reply of the complainant, and the dialog is finished when the argumentation reaches a stable state. The questions can be selected dynamically, such that the argumentation advances towards a stable state with each question. This approach is similar to the current practice for reporting a crime at a police station, where a police officer asks a number of questions in order to fill out a crime report.

Note that, for practical purposes, the two approaches can be considered as opposite ends of the same methodology, i.e., providing a complete document to the argumentation graph is essentially a dialog with a single user response. Similarly, a question within a multi-step dialog can result in a complex user response which can be considered a short document. Regardless of the length of the dialog, the answers need to be parsed and processed in order to extract relevant information. This could be avoided by using closed-form questions with a list of predefined answers (e.g., 'Did you receive a package?', 'How long did you wait?'), however such a dialog may prove to be insufficient for users to explain the details of the situation.

4 QUESTION POLICY LEARNING

When using a dialog between the system and the complainant, we want the system to get to a stable state as efficiently as possible. Determining whether a state is stable consists of hypothesizing over all possible future questions. As this is generally infeasible to do, we turn to machine learning methods to train a questioning strategy to approximate an ideal solution.

The policy that is to be learned maps observed propositions to questions that can be asked or to a terminating action (accept/reject). The action results in some response from the user, which consists of new observations and possibly inferred conclusions (both propositions) that are added to the already known propositions. As a result, we may view the state of the system as a set of propositions and the actions as non-deterministic transitions between states. If we model this as a Markov Decision Process, then we can use Q-learning [9] to train a policy. Note that this requires the assumption that the answer to a question is independent of how the current set of propositions is obtained. For our Q-learning approach we require a reward function. In order to promote efficient dialogs, we give a small penalty for each action. To promote stability, we give a high reward for reaching stable states. Finally, alongside a reward function we need a user model that realistically provides responses to questions (the probabilities of transitions in the Markov Decision Process). To this end we currently work with handwritten models. When the system is deployed it will gather user data and then a data-driven model will replace the initial model.

As an example, using the argumentation graph in Figure 1, consider the state in which *false location* is known to be true and all other propositions are unknown. Suppose the Q-learning algorithm selects 'ask for *false website*' as the next action to evaluate. This question can support *deception* as a conclusion, however this conclusion was already supported by *false location*. The new state after asking this question therefore has the same reward value as the previous state, while the penalty is increased by performing the question action. This will lead the Q-learning algorithm to reject this state-action pair as part of the policy, and to consider alternative actions instead.

5 EXTRACTING ENTITIES AND RELATIONS

As described in Section 3, user input (either from report documents or from dialog responses) needs to be mapped to propositions in the argumentation graph. These propositions generally consist of a relation between relevant entities (people, objects, locations,

[1] Stability is not fully calculated (due to computational complexity). Instead, we deploy a heuristic that runs polynomial in the number of argument graph edges.

etc.) described in the complaint. Various techniques can be used to extract entities and relations between entities from text, ranging from dictionary lists and syntactic patterns to complex parsing algorithms and machine learning models. For Dutch legal data the Dutch dependency parser Frog [2] can be used for named entity recognition, for which the performance on legal data is evaluated in previous work ([8]). For relation extraction the development and evaluation of automatic methods is an ongoing effort in the current research project, as described in the remainder of this section.

In order to use these techniques effectively in a law enforcement application, the expected result from text processing should be considered carefully. Given the domain, for example, knowing whether the victim has paid the counterparty is essential. However, other information containing entities (e.g., details of contacts with other victims) are not relevant for legal reasoning. The relevance of certain types of information determines how data should be collected and processed in developing entity and relation extraction methods. This includes a mapping from nodes in the argumentation graph to entities and relations in the text. However, other propositions may not be represented directly in the text, such as the use of a false website. In such cases, partial information may be present in the text (e.g., the counterparty operated a website), while the proposition itself can only be validated after considering information from a third party (e.g. checking with the ISP to prove that the website is fake). However, in both cases the legal definition of the crime (as expressed in the argumentation graph) is essential for the development of text processing methods.

6 DATA ANNOTATION FOR RELATION EXTRACTION

As we stated in Section 5, some of the propositions in the argumentation graph are based on relations between entities in the crime report documents. We plan to use supervised machine learning techniques (see for example [10]) to automatically extract these relations, which has shown to provide high accuracy for the current dataset in preliminary experiments. Therefore, crime report documents need to be annotated with the concepts identified in the domain analysis process. Concepts of interest include residence, payment and delivery information. Each concept has a number of associated properties for which annotation could prove useful. These properties are listed in Table 1.

Residence relations are interesting as they may indicate the deceptive trick in which the fraudster gives a false address. In that case, we often find in the report that the actual occupant of the address was an unrelated person who did not know anything about the advertisement. Furthermore, the address is often in a remote relation, facilitating the fraudster to (falsely) promise to send items per mail. To be able to detect these situations in the future, we annotate the person name, location, role of the person and large distance property. In future research the processing of coreferential expressions (e.g., the token *he* to refer to an earlier mention of *John Smith* in a document or dialog) will be addressed.

Payment and delivery information are captured by send and receive relations. The reason for this is that the complainant usually only knows one side of the story: if the complainant intended to buy a product, he or she typically claims having sent money to the counterparty without having received the product. We do not know for sure if the counterparty received the money and/or sent the product, as there is a possibility of a delivery or payment failure by a third party. The send and receive relations are ternary, having a sender, receiver and object, although some of the entities may be omitted in the text: for instance, in the sentence 'I did not receive anything' the sender is missing. For the sender and receiver, we annotate the corresponding character indices and the role (complainant, counterparty or other). In some complaints, the complainant reports that he or she received a broken product. This suggests a civil case instead of a fraud case. Therefore, we annotate not only the character indices but also the state of the product. Furthermore, we annotate the word(s) indicating a send or receive relation and the validity of the relation. An example annotated sentence (translated for illustration purposes) is provided in Figure 2.

concept	*property*
residence	name of person
	location
	role: complainant, counterparty, related or unrelated
	large distance
send	sender
	recipient
	object
	indicator of relation
	validity: true, false or unclear
	type: product, payment, contact or other
	role sender: complainant, counterparty or other
	role recipient: complainant, counterparty or other
	state object: fake, broken or other
receive	sender
	recipient
	object
	indicator of relation
	validity: true, false or unclear
	type: product, payment, contact or other
	role sender: complainant, counterparty or other
	role recipient: complainant, counterparty or other
	state object: fake, broken or other

Table 1: Examples of properties of interest for annotation.

We have transferred €100,- to this man on account number 1234.

Relation:	send
Sender:	*we*
Role:	complainant
Recipient:	*this man*
Role:	counterparty
Object:	€100,-
Indicator:	*transferred*
Status:	sent (other options: 'not sent' and 'unclear')

Figure 2: Example annotated sentence.

We plan to use the annotations in a classifier that, given a set of tokens, decides if they are entities in one of the aforementioned relations. The output of this classifier can then be mapped to propositions for the argumentation graph. Such a classifier is intended to operate on free text input, using simple features such as the presence of selected keywords as well as more complex features such as lemmas or grammatical dependency paths. For real-world free text the computation of these features may be unreliable (e.g., as a result of misspellings in the source text) or, even with reliable features, a real-world example may not conform to the regularities found in the training set. However, using a suitable classifier and an appropriate training set size, the model is expected to generalize over irregularities to a certain extent. Moreover, as mentioned in Section 3, using a dialog component within the system will provide some context to interpret the results of the relation classifier.

7 CONCLUSION

In this paper we have described an approach to automatically handling the intake of criminal reports filed online by citizens. The proposed approach combines different types of techniques (i.e., natural language processing, argumentation and Q-learning) to obtain a system that A) handles natural language, B) produces arguments for complex conclusions and hence provides understandable and legally sensible explanations for decisions regarding complaint reports, and C) is capable of gathering information from its environment efficiently by only asking the most relevant questions to the user and terminating the process if no more relevant information is to be found.

The algorithms ans implementations presented in this paper are currently under development and a number of prototypes are working or nearing completion. Furthermore, parts of the system, such as automatically drawing conclusions using the argumentation graph, the named entity recognition and basic relation extraction, have been implemented in the existing development systems at the Dutch National Police.

The techniques developed are generalizable beyond the domain of online trade fraud. Extending the system to other domains will involve a substantial (knowledge) engineering effort: argumentation theories will have to be built for different domains, and algorithms for extracting new types of observations will have to be trained. While our solution thus suffers from the classical “knowledge engineering bottleneck” that has hampered knowledge-based systems for decades, we believe the focus on smaller, relatively simple assessments makes true autonomous systems more feasible. Furthermore, building and maintaining a small argumentation theory may be more suitable for general IT personnel at the Dutch Police than training machine learning algorithms on a new dataset. Finally, the algorithms for entity and relation extraction are aimed to be as general as possible, with good performance in different domains. Thus, other tasks and processes within the police organization can be gradually incorporated into the framework.

REFERENCES

[1] Jeroen Bergers. 2018. *Improving online trade fraud complaint handling using argumentation theory.* BSc. Thesis. Utrecht University.

[2] Antal van den Bosch, Bertjan Busser, Sander Canisius, and Walter Daelemans. 2007. An efficient memory-based morphosyntactic tagger and parser for Dutch. In *Selected Papers of the 17th Computational Linguistics in the Netherlands Meeting.* Netherlands Graduate School of Linguistics, 99–114.

[3] Phan Minh Dung. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence* 77 (1995), 321–357.

[4] Ilse van 't Hul. 2018. *Improving online trade fraud complaint classification by applying machine learning techniques.* BSc. Thesis. Utrecht University.

[5] William Kos, Marijn Schraagen, Matthieu Brinkhuis, and Floris Bex. 2017. Classification in a Skewed Online Trade Fraud Complaint Corpus. In *Preproceedings of the 29th Benelux Conference on Artificial Intelligence.* 172–183.

[6] Henry Prakken. 2010. An abstract framework for argumentation with structured arguments. *Argument & Computation* 1, 2 (2010), 93–124.

[7] Henry Prakken and Giovanni Sartor. 1996. A Dialectical Model of Assessing Conflicting Arguments in Legal Reasoning. *Artificial Intelligence and Law* 4 (1996), 331–368.

[8] Marijn Schraagen, Matthieu Brinkhuis, and Floris Bex. 2017. Evaluation of Named Entity Recognition in Dutch online criminal complaints. *Computational Linguistics in The Netherlands Journal* 7 (2017), 3–16.

[9] Chris Watkins. 1989. *Learning from Delayed Rewards.* Ph.D. Dissertation. King's College London.

[10] Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths. In *Proceedings of EMNLP 2015.* Association for Computational Linguistics, 1785–1794.