Extraction of semantic relations in noisy user-generated law enforcement data

Marijn Schraagen
Institute for Information and Computing Sciences
Utrecht University, The Netherlands
M.P.Schraagen@uu.nl

Floris Bex
Institute for Information and Computing Sciences
Utrecht University, The Netherlands
F.J.Bex@uu.nl

***Abstract*—Relation extraction from text is a well-known and extensively studied topic in Natural Language Processing research. However, the implementation of relation extraction approaches in real-world application scenarios raises various methodological considerations which are often left implicit in existing research. This paper explores these considerations using a real-world dataset of user-generated police reports in Dutch. The use of linguistic features based on dependency trees is investigated, including an ablation analysis of the importance of individual features. The construction of negative examples for machine learning models is discussed, as well as the construction of a baseline model. The methodological implications of using a small dataset are discussed in terms of the design and performance of a Long Short Term Memory network as well as a Support Vector Machine. In general the models perform well, however the definition of the classification task, and in particular the construction of negative examples, are shown to have a large impact on classification accuracy and subsequently on the interpretation of the evaluation results.**

I. Introduction

Relation extraction, defined as *finding instantiations of specific semantic relations between entities in text*, has applications in various text processing tasks such as summarization, reasoning, and sentiment analysis. This paper addresses methodological considerations for the relation extraction task based on a case study of Dutch web form data in the legal domain. Although relation extraction has been investigated extensively in previous research, the implementation of this research in a real application setting raises several considerations regarding the definition of the task and evaluation of the performance. In this paper the following issues are discussed:

1) Section III: which task is addressed by a relation extraction approach? What are the prerequisites for this task?
2) Sections IV,VII: are dependency tree features useful for relation extraction in noisy user-generated data?
    a) Is the performance of state-of-the-art tools for Dutch on this type of data sufficient to extract features from data?
    b) Section VII-A1: which features are most useful for this task?

    Note: the term *dependency tree features* refers to the sequence of token-level features generated for a particular pair of tokens by computing the dependency tree for a sentence and extracting the dependency path between the tokens. This procedure and the dependency parser used in the experiments are described in Section IV.
3) Section V: which methods can be used for realistic negative sampling of relation examples?
4) Section VI: what is a good baseline for this task?
5) Section VII: how do standard machine learning algorithms behave for a small amount of data?
    a) Which architecture can be used for different machine learning methods?

> I would like to report fraud. I recently saw a bicycle for sale at an online trading platform and contacted advertiser John Doe. Because he said that he lived in a different part of the country, we agreed that he would send the bicycle to my home address in Amsterdam. I paid him in good faith but still have not received the bike. Mr. Doe does not respond to my e-mails any more. I did some research and saw that there is a second user called John Doe, but that person claims to live in Amsterdam and uses another account number.

Fig. 1. Example document from the crime report dataset with relation annotations (adapted, translated and anonymized for presentation purposes).

For the case study manual annotation has been performed on a dataset for a single type of relation, i.e., residency (*John lives in Miami*). An example document is shown in Figure 1. Using the results of the current research, other types of relations can be added in future work. The issues investigated in this paper are centered around the specific nature of the dataset, i.e., unmoderated, user-generated, noisy, short texts, in Dutch, in the legal domain. This research therefore explicitly does not aim to provide a general comparison between relation extraction methods, or to benchmark the performance of methodological adjustments using well-established datasets such as MUC or ACE. Instead, the research aims to provide an overview of issues and methodological aspects that need to be addressed for the task of relation extraction on datasets containing this type of data, and to provide a case study for this task using well-known machine learning algorithms.

II. Related work

The task of relation extraction has been studied extensively as a subtask of information extraction. Early approaches used domain-specific pattern matching [1], [2]. Facilitated by shared tasks such as MUC [3], ACE [4] and SemEval [5], domain-independent approaches have been developed using statistical machine learning. Models have been trained using shallow linguistic features such as suffixes, capitalization or sentence position of words [6] or POS features [7]. Based on the assumption that semantic relations are correlated with syntactic structure, various approaches are developed using dependency relations as a feature. Different algorithms have been applied, such as Maximum Entropy models [8], Support Vector Machines [9], [10], and more recently neural network models using word embeddings, such as convolutional neural networks [11] and Long Short Term Memory networks [12], [13], [14]. Performance comparisons (in, e.g., [12] and [15]) show that, although neural network approaches generally achieve high accuracy, the selection of features has a relatively large influence on model accuracy compared to the selection of the learning algorithm. Moreover, neural networks are assumed to require a relatively large training set for convergence compared to other machine learning algorithms [16], [17], [18], parameter settings for complex networks are non-trivial, and computational cost may be prohibitive. Furthermore, using word embedding features requires either a set of pre-trained vectors for the target language, or an additional network layer for training custom vectors within the dataset which may increase the amount of data necessary for convergence. As one of the goals of the current research, this issue is investigated in more detail.

III. Task definition

For supervised classification approaches a training set needs to contain examples for all predefined classes. In commonly used datasets generally several types of relations are annotated (e.g., SemEval 2010 contains 10 different relation types such as *cause–effect, component–whole, message–topic*). This means that a positive example for a particular relation can be used as negative example for other relations. However, in case the dataset contains only examples of a single class (such as the current case study, see Section IV), the relation extraction task needs to be extended with a methodology to construct explicit negative examples to provide as training data. The task of constructing negative examples is non-trivial, given that the negative examples should be similar to the positive examples in order to train a practically useful classifier, while the negative examples should not be too similar, which would prevent a classifier from finding a decision boundary. In Section V methods for constructing negative examples are discussed in detail.

A further issue relates to the general scope of the relation extraction task. Existing approaches generally consider the task of classifying given pairs of entities. However, the selection of entity pairs from a document to present to the classifier (i.e., the information retrieval aspect) is usually not addressed by existing approaches. This issue can be irrelevant for a number of reasons, e.g., the entities can be trivially produced from a list (such as protein or drug names), or by an algorithm for named entity recognition, or given as a query by a human user, or simply assumed to be given in a research context. However, for real-world applications, entities may not be known in advance and the identification of potential entities may not be trivial, for example because the entities are not necessarily named entities (e.g., *the man lives near the country border*). The task definition therefore changes from classifying predetermined pairs of entities to identifying and classifying candidate relation pairs in non-annotated text. Note that the retrieval aspect has been investigated in closely related research areas such as temporal information processing [19]. Results from such tasks show that the retrieval aspect is indeed a significant component of the task, with F1-scores ranging from around 0.3 on raw text to around 0.5 when candidate pairs are given.

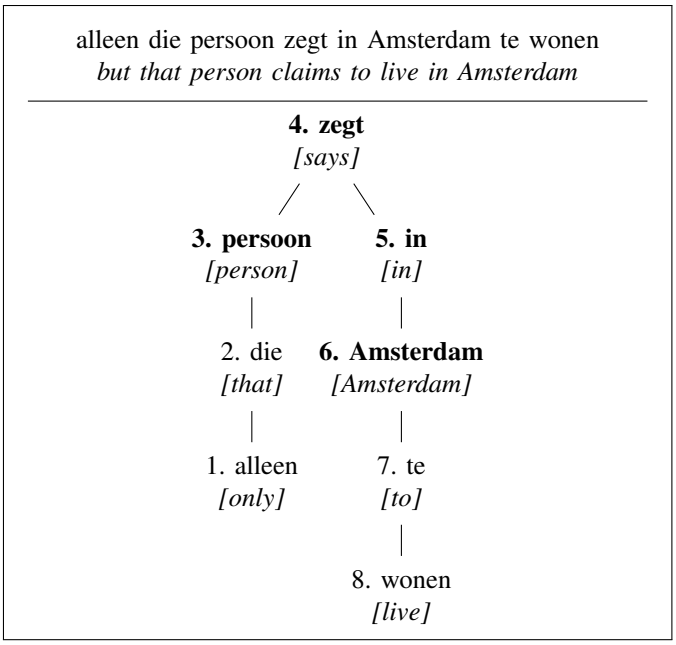

Fig. 2. Dependency tree for a (partial) example sentence, containing the residency relation *(person,Amsterdam)*.

The issue of identifying elements of a relation can be addressed by generating candidates from text. A model trained on these candidates (and evaluated using classification accuracy on a balanced test set) can be applied on a new document by generating all candidates in that document and classifying each candidate. The number of candidates can become very large, which is both a practical issue as well as a theoretical problem in selecting representative negative examples for training. A restrictive heuristic can be applied to limit the number of candidates, as detailed in Section V. A further discussion of the use of a selection heuristic for training examples is provided in Section VIII.

translation	only	that	**person**	**says**	**in**	**Amsterdam**	to	live
surface	alleen	die	persoon	zegt	in	Amsterdam	te	wonen
lemma	alleen	die	persoon	zeggen	in	Amsterdam	te	wonen
full POS	ADV()	PRON(sg,demonstr,nom)	N(sg,common,gendered)	V(pres,sg)	PREP(init)	SPEC(name)	PREP(init)	V(inf)
main POS	adverb	pronoun	noun	verb	preposition	special	preposition	verb
named entity	-	-	-	-	-	location	-	-
chunk	ADVP	NP	NP	VP	PP	NP	VP	VP
direction	→	→	→	←	←	←	←	←

TABLE I
TOKEN FEATURES FOR THE EXAMPLE SENTENCE IN FIGURE 2.

alleen	die	persoon	zegt	in	Amsterdam	te	wonen		en	geeft	een	ander	rekeningnummer		
only	that	person	says	in	Amsterdam	to	live		and	gives	an	other	account_number		
advp	{	np	} vp	pp	np	{	vp	}	conj	vp	{		np	}	

but that person claims to live in Amsterdam and provides another account number

candidates N_1 for dependency path length 4 (11 in total)	only–says, that–in, that–and, **person–Amsterdam**, person–gives, says–to, says–account_number, in–live, in–gives, and–in, and–other
relation marker	live (wonen)
allowed chunk types	NP, ADVP, ADJP, SBAR, unlabeled
candidates N_2, window size 5 (20 in total)	**person–Amsterdam**, Amsterdam–person, person–an, an–person, ..., other–account_number, account_number–other

Fig. 3. Example generation using methods N_1 and N_2. A positive training example is shown in bold, all other pairs are negative examples.

IV. Case study

A set of user-submitted crime reports in Dutch is used as a case study, as part of the project *Intelligence Amplification for Cybercrime* [20]. Reports are generated from a web form containing various specific fields (e.g., names, addresses, bank account numbers) and a free entry text field intended for a description of the situation. All reports are in the domain of internet fraud (e.g., an item is purchased online but not delivered). The case study focuses on the automatic extraction of *residency* relations from the description text, since this is often relevant for a police investigation. Using manual annotation by a single annotator, a gold standard set of 2975 relation occurrences has been collected. Annotations consist of a part of a sentence containing a subject token, an object token and all tokens in between (see Figure 1 for examples). The Dutch dependency parser *Frog* [21] has been applied on all documents to extract various linguistic features, including dependency paths between relation elements. In Figure 2 and Table I the list of features is shown, as applied to an example phrase representing a residency relation. These features are used to predict instances of residency relations in the corpus. Note that the list of features includes both *full Part-of-Speech* and *main Part-of-Speech*. The Frog parser uses the CGN tagset [22] which defines ten main tags, e.g., *noun, verb, pronoun*, and a large set of (mostly morphosyntactic) properties for each tag, e.g., *singular, diminutive, past tense*. In the current experiments the POS tags are used both with and without the additional properties, as *main POS* and *full POS*, respectively.

V. Development of training data

The dataset in the case study contains a single relation (i.e., residency), therefore explicit negative examples need to be constructed. In order to create a representative training set (i.e., to obtain a decision boundary in a trained model which is close to the actual boundary of the positive class), the negative examples need to be sufficiently similar to the positive examples, while not being too similar to obscure the decision boundary.

In the current experiments, two approaches for the construction of negative examples have been tested. In the first approach (N_1) a training set is developed by pairing each positive example with a random phrase for which the dependency path has the same length as the positive example. Negative examples are extracted from a random document in the corpus, i.e., positive and negative examples are not semantically related. Moreover, syntactically the only similarity between examples is the length of the dependency path, without restrictions on parts-of-speech or surface form. In Figure 3 a number of generated candidates is shown for an example sentence.

In the second approach (N_2) a heuristic is designed to extract candidate relation instances from the corpus. This heuristic defines a relation marker token, a restricted set of chunk types for relation elements, and the size of a window around the relation marker where relation elements can be located (see Figure 3 for an example). The relation marker tokens as well as the chunk types have been determined man-

ually[1] based on analysis of the set of positive examples. This heuristic produces a large number of candidates, of which, by design, a large majority is not an actual relation instance. From these candidates a training set is developed using all positive examples from the manually annotated dataset that satisfy the heuristic, and a random selection (of equal size) of negative examples that satisfy the heuristic. Considering the sentence in Figure 3, for example, the pair *person-Amsterdam* is included in set N_2 as a positive example, while one or more of the other candidates (such as *person-an* or *account_number-other*) could be randomly selected as negative examples in set N_2. Because the heuristic is restrictive, 28% of actual, annotated positive examples do not satisfy the criteria (e.g., when one of the entities is part of a VP chunk) and are therefore excluded from the training set in order to retain similarity between positive and negative examples.

The currently used dataset contains manually annotated positive examples, allowing for supervised machine learning. Because of the presence of these annotations, there is no need to generate positive examples automatically. However, note that both methods for generating negative examples are also capable of generating positive examples, either by randomly extracting dependency sequences of length n (method N_1) or by applying a more restrictive heuristic (method N_2). This entails that these methods of generating training data, as well as classifiers trained using this data, can be used to generate and classify candidate pairs of relation entities in unseen data without annotations. As noted in Section III, this task definition differs from previous approaches using data from, e.g., MUC or SemEval, that provide explicit candidate pairs for both training and evaluation. Instead, the current methodology is more similar to information retrieval tasks, where the goal is to identify relevant units of information in a larger document or corpus. In particular the approach using a linguistic heuristic (N_2) shows similarity to information retrieval, because the heuristic restricts the set of candidates in such a way that also a number of positive examples is excluded from the candidate set. This shift in task definition, and the implications for the current approach, will be discussed in Section VIII.

VI. Baseline

A typical pattern for residency relations consists of the sequence *[person] lives in [location]*. This pattern is used to create a baseline classifier which is intended as an indication of the added value of machine learning models for this task. The baseline pattern is a sequence of the following elements:

1) A noun phrase (as determined by the parser)
2) 0–3 arbitrary tokens
3) Preposition *in*, *from* or *at* (Dutch: *in, uit, te*)
4) A location-like token, either starting with an upper case letter or appearing on a list of Dutch location names[2].

[1]Marker tokens are defined using a regular expression for the word *live*, allowing for spelling variants as well as grammatical variants (verb, adjective, adverb). See Section VIII for a discussion on the use of marker tokens.

[2]https://nl.wikipedia.org/wiki/Lijst_van_Nederlandse_plaatsen

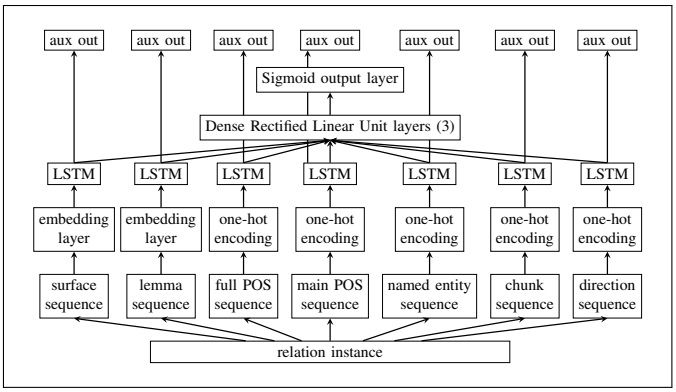

Fig. 4. LSTM network architecture.

sequence length	21 tokens
input data type	integer (token indices)
vocabulary size	
surface	4938
lemma	4151
embedding layer	
vector length	512
mask zero	true
LSTM layer	
units	32
merge	concatenate
auxiliary output	
type	dense
activation	sigmoid
hidden layers	
type	dense
output units	64
activation	rectified linear unit
number of layers	3
output layer	
type	dense
activation	sigmoid
training	
optimizer	root mean square propagation
loss function	binary cross entropy
all other parameters	*Keras defaults*

TABLE II
PARAMETER VALUES USED FOR THE LSTM MODEL.

Note that the definition of a pattern-matching baseline is necessarily arbitrary. However, the current baseline significantly outperforms a fully random or fixed class assignment (which would both produce an accuracy of 0.5 for a balanced two-class dataset), indicating that this baseline indeed captures a nontrivial amount of domain knowledge necessary for the extraction of residency relations.

The baseline pattern could be extended to improve recall, for example by extending the subject element to other chunk types or by allowing tokens between the preposition and the location element. However, such extensions would create an increasingly non-trivial pattern. The goal of the current baseline is to show that the problem of extracting residency relations from text is in fact non-trivial, and therefore a machine learning approach is appropriate for this problem.

VII. Experimental evaluation

To test the performance of complex machine learning models on small amounts of data, a Long Short Term Memory network (Section VII-A) as well as a Support Vector Machine (Section VII-B) are evaluated on the crime report dataset.

A. Long Short Term Memory network

The performance of a Long Short Term Memory model (LSTM) is tested in a series of experiments. Note that this implementation is not intended as a benchmark evaluation of the model or as modification of network architectures from related work. Instead, the experiments serve as a proof-of-concept implementation of the issues discussed in previous sections, i.e., approaches to feature extraction, construction of training examples and a baseline model for the application of machine learning on a small user-generated dataset.

The network setup is similar to [12], using parallel LSTM layers for each of the seven features (see Figure 4). The *surface* and *lemma* features are converted to an embedding representation, while the other features are represented using one-hot encoding. The LSTM classifier is trained using the Keras framework with TensorFlow backend (see https://keras.io), using parameter values as listed in Table II.

The results of training the LSTM network are listed in Table III. Two separate models L_1 and L_2 have been trained on datasets N_1 and N_2, respectively. For comparison, model L_1 is also tested using dataset N_2, as well as model L_2 using dataset N_1. The results show that the negative examples in set N_2 are more difficult to train than the example in set N_1. This is also indicated by cross-model performance.

In set N_1 the negative examples have been randomly selected from the corpus, with the length of the dependency sequence as only constraint. As a consequence, in general the negative examples are syntactically and semantically highly different compared to the positive examples. These large differences can be easily identified using a machine learning algorithm, which is shown by the score of 0.99 for dataset N_1 in Table III. However, such a classifier is less capable of classifying more realistic examples correctly, as shown by the performance of classifier L_1 on set N_2. Therefore, dataset N_1 is considered non-representative for the relation extraction task. On dataset N_2 the LSTM model results in an accuracy of 0.95 (as highlighted in Table III). The implications of this issue will be discussed further in Section VIII.

Given the small amount of data and the high accuracy scores, an analysis of the errors is difficult to generalize. However, manual inspection of the errors has shown that misclassified positive examples often contain a relatively more complex syntactic structure, such as coordinating conjunctions (*and he lived in Amsterdam*) or complex noun phrases (*the landlord of the apartment as advertised online lives in London*). Conversely, several misclassified negative examples closely resemble a positive example (*she is in Amsterdam*) or consist of a partial residency relation (*the woman from the advertisement lived*). It is not always clear however why a particular error is made. Increasing the sample size may improve this type of analysis, both by reducing the number of accidental classification errors as well as by enabling a more robust quantitative error analysis approach.

classifier	*dataset*	pos	neg	all
baseline	N_1	0.36	0.99	0.68
baseline	N_2	0.50	0.92	0.71
L_1	N_1	0.99	0.99	0.99
L_2	N_2	**0.92**	**0.97**	**0.95**
L_1	N_2	0.99	0.75	0.87
L_2	N_1	0.93	0.97	0.95

TABLE III
LSTM classification accuracy using 10-fold cross validation.

rank	*accuracy*	*combination*
1.	0.961133583	full POS, main POS, named entity, chunk type, direction
2.	0.959109304	full POS, main POS, named entity, chunk type
3.	0.952631573	full POS, main POS, chunk type, direction
4.	**0.94777328**	**surface, lemma, full POS, main POS, named entity, chunk type, direction**
5.	0.944534425	full POS, main POS, named entity, direction
6.	0.943522279	surface, lemma, full POS, main POS, chunk type, direction
7.	0.943117426	surface, lemma, full POS, main POS, named entity, chunk type
8.	0.938259124	surface, lemma, full POS, main POS, chunk type
9.	0.93684211	surface, lemma, full POS, named entity, chunk type
10.	0.93663969	surface, lemma, full POS, named entity, chunk type, direction
24.	0.920850203	full POS, main POS
63.	0.891902833	surface, lemma
108.	0.857894746	full POS
114.	0.851012152	surface
118.	0.838663973	main POS
119.	0.835222674	lemma
120.	0.831376525	full POS, named entity, direction
121.	0.812955458	chunk type
122.	0.810526303	named entity, direction
123.	0.78259109	main POS, named entity, direction
124.	0.781578948	main POS, chunk type, direction
125.	0.730364364	named entity
126.	0.727327936	direction
127.	0.721255062	named entity, chunk type, direction

TABLE IV
Selection of ablation results.

1) Feature ablation: As further analysis of the results an ablation experiment has been performed with all combinations of features (127 feature combinations in total for 7 features). This experiment is performed using 10-fold cross-validation on the L_2, N_2 setup, which is considered the most realistic scenario. A selection of results is shown in Table IV.

The results show that the *full POS* feature is highly informative for relation classification. This feature is present in the 10 highest scoring models (with accuracy values ranging from 0.93–0.96), and it is also the feature that performs best in isolation (accuracy 0.86). In contrast, the *surface* and *lemma* features do not perform as well as expected. Even though these features occur in the highest scoring models, they are generally complemented with a POS feature (*full POS*, *main POS*, or

both). In isolation these features result in lower accuracy compared to the POS features, showing differences for the single features (*full POS* 0.86, *main POS* 0.85, *surface* 0.85, *lemma* 0.84) and when combined ({*full POS, main POS*} 0.92, {*surface, lemma*} 0.89). A possible cause for this behavior is the relative complexity of the *surface* and *lemma* features, with a large vocabulary size and an embedding layer, as opposed to the *full POS* feature (148 possible values) or the *main POS* feature (14 possible values), both presented to the network in one-hot encoding. Another possible cause is that vocabulary items may be used in various configurations independent of the presence of a relation, while the grammatical structure as represented in a POS sequence is more directly correlated to relation occurrences.

Less useful features include *direction* and *chunk type*, which is as expected given the coarse-grained nature of these features. The *named entity* feature can be useful, however this is mostly observed in combination with other features. This can be explained by the sparse nature of the named entity feature, which indicates that the presence of the feature can be informative, while the absence of the feature does not influence the classification to a large degree.

Using all available features results in an accuracy of 0.95 (rank 4), which indicates that the full set of features is indeed useful for the classification of relation instances.

B. Support Vector Machine

The LSTM model allows inputs to be presented to the network as a sequence. This reduces the number of input nodes required to represent sequence inputs, given that one input can be used for the full sequence. In contrast, non-recurrent models such as Support Vector Machines (SVMs) need to process each element of an input sequence separately. For longer sequences this leads to a very extensive input layer, which means that both more parameters need to be trained and the sparsity of the input increases. To address this issue, the dimensionality of the input can be reduced during preprocessing. For SVMs this can be achieved by defining a custom kernel function that computes similarity between input sequences (cf. [10]), reducing the dataset to a collection of single-valued similarities between examples. In the current experiments similarity is defined as the number of common features for each pair of tokens for equal-length phrases, and 0 otherwise (i.e., $sim(A,B) = \sum_i \sum_{f \in features} A_i(f) = B_i(f)$ if$||A|| = ||B||$, else 0).

The dependency kernel is non-normalized, i.e., ranging from 0 to $L \cdot F$ for path length L and number of features per token F. As a consequence, the maximum score is higher for longer paths, which may influence classification accuracy. Two normalization approaches have been tested, i.e., $sim_{exp}(A,B) = 1 - e^{-\gamma sim(A,B)}$ for various values of γ, and $sim_{prop}(A,B) = \frac{sim(A,B)}{||A|| \cdot F}$. Note that sim_{exp} (see Figure 5) effectively maps high similarity scores to 1 (i.e., path length is not taken into account, as in the original kernel), whereas sim_{prop} is a proportional scaling function.

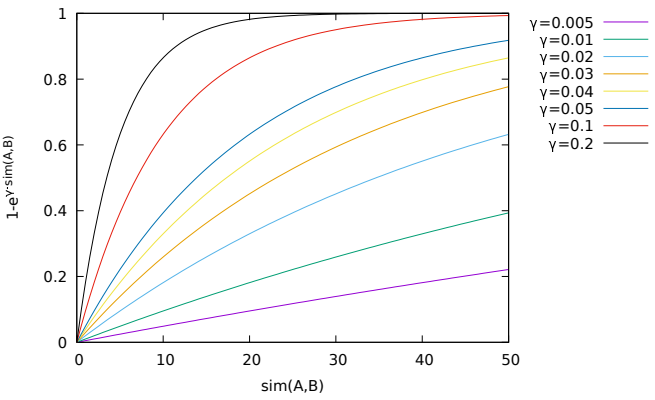

Fig. 5. Behavior of a SVM scaling function using various γ values.

classifier	*dataset*	pos	neg	all
baseline	N_1	0.36	0.99	0.68
baseline	N_2	0.50	0.92	0.71
S_1	N_1	0.95	0.95	0.95
S_2	N_2	0.89	0.89	0.89
S_1	N_2	0.99	0.52	0.76
S_2	N_1	0.85	0.93	0.89

TABLE V
SVM CLASSIFICATION ACCURACY USING 10-FOLD CROSS VALIDATION.

The results of the experiments are listed in Table V. As for the LSTM model, the baseline is compared to a Support Vector Machine S_1 trained using dataset N_1 and a second SVM S_2 trained using the dataset N_2. Evaluation has been performed both within and across datasets. Similar to the LSTM results, the negative examples in set N_2 are more difficult to train (using the current dependency kernel). The cross-model performance shows that the accuracy of model S_2 is similar on both sets, however model S_1 is unable to classify set N_2 correctly (i.e., accuracy on negative examples is at near chance level).

The influence of scaling for the SVM classifier is tested using S_2, N_2. However, both exponential scaling and proportional scaling result in lower accuracy compared to the unscaled kernel.

VIII. DISCUSSION

The results show that the LSTM model outperforms the SVM model in all combinations of training and test data. This indicates that the LSTM model, while being relatively complex, is still capable of generalizing over a small dataset. The performance of the SVM could be influenced by the model itself, however the dimensionality reduction method using a distance kernel may also explain the difference with the LSTM model. Furthermore, scaling functions for dependency path comparisons are shown to negatively impact model accuracy.

Considering the training data, the results show that the selection of negative examples has a large influence on model performance. The examples in set N_2 can be considered to be more representative as relation candidates, therefore the

accuracy of the models using this dataset offer a more realistic evaluation compared to the models trained on set N_1.

The word embeddings are trained within the network, using dependency token sequences as input. Alternatively, existing embeddings for Dutch could have been used (e.g., [23]). However, pre-trained embeddings from a general domain are often less useful for specific tasks (see, e.g., [24], [25], cf. also [12] on which the design of the LSTM in this paper is based). Given the current results the non-pretrained embeddings work well, which was somewhat unexpected given the limited amount of data (around 30k tokens in total). Further interpretation of this conclusion should consider the results of the ablation experiment (Section VII-A1), which shows that the embedding features are outperformed by the POS features represented with one-hot encoding. Nonetheless, the ablation results also show that the embedding features trained on the current small dataset do contribute positively to classification accuracy.

In the current research a heuristic is used to identify a set of potential training and test examples, from which the actual training and test data is sampled (see Section V). This heuristic needs to be defined such that the resulting examples are realistic potential candidates, without being too restrictive. The precision of the heuristic is expected to be low (e.g., in Figure 3 only one out of 20 candidates is a positive example), but the generated negative examples should still somewhat resemble the positive examples. More importantly, the recall of the heuristic should be high, in order to include as many valid positive examples as possible in the training and test data. This balance is difficult to achieve, the heuristic used in the current case study for example includes only 72% of the positive examples in the annotated corpus. The task definition therefore changes, from identifying all relation occurrences in a text to identifying relation occurrences that satisfy the heuristic. This can be a significant restriction, such as in the case of the heuristic used in the current research, which is defined in terms of a small set of relation marker tokens and grammatical chunk categories. However, the practical and methodological implications of this issue are not trivially clear, and need to be investigated in more detail in further research.

To summarize the contributions of the current paper, the methodological issues stated in Section I are addressed below.

1) *Which task is addressed by a relation extraction approach? What are the prerequisites for this task?*

Relation extraction can be defined as a classification task, i.e., to determine the type of relation between two given entities. However, in many application scenarios the scope of the task needs to be extended to include a retrieval aspect, i.e., given a text, determine which tokens are candidate elements of a semantic relation, and determine the type of relation once the candidates have been identified. For this task definition a methodology for generating candidates needs to be developed. In the current paper two approaches have been investigated, using various linguistically motivated criteria for selecting relation candidates. In the current case study this approach has been used to create negative examples for training a classifier, which was necessary given that the annotated dataset contains only a single type of relation. In case multiple types of relations are available it is not necessary to construct negative examples. However, in order to extend the task definition from classification to retrieval, also in the case of multiple relation types a candidate generation methodology is needed.

2) *Are dependency tree features useful for relation extraction in noisy user-generated data?*

Dependency tree features can be used successfully for this type of data, even when performing strong dimension reduction.

a) *Is the performance of state-of-the-art tools for Dutch on this type of data sufficient to extract features from data?*

The statistical dependency parser Frog provides sufficiently robust linguistic features for this type of data. The parser acts as a filter on various errors in user-generated text, e.g., incomplete sentences, meta-characters introduced by copying fragments of e-mail conversations or website content, spelling errors or grammatical errors. This type of data does introduce errors in sentence splitting, lemmatization, POS tagging, dependency parsing, etc., however a statistical parser is generally capable of producing a reasonable linguistic analysis. Combined with the data generalization performed by the classifier, this type of data does not significantly impair classification accuracy. It should be noted, however, that the phrases containing relation examples are contain a relatively low amount of noise. If an NLP task would be performed using different parts of the text, then data quality may prove to be a more problematic issue.

b) *Which features are most useful for this task?*

The ablation experiments show that Part-of-Speech features are highly informative for the relation extraction task. Vocabulary features (surface form and lemma) contribute less to the overall task accuracy, however this may be caused by the relatively small amount of training data available for the embedding layers.

3) *Which methods can be used for realistic negative sampling of relation examples?*

Using a heuristic that imposes linguistic constraints on constructed negative examples creates a more complex classification problem, which is therefore more useful in practise. However, the resulting accuracy score (0.95) still suggests that the set of negative examples should be restricted further. An undesired side effect of such a restriction may be, however, that a larger number of actual positive examples will also be excluded from training. Alternatively, the task of residency relation extraction as defined in the case study might be insufficiently complex itself. Related work on SemEval 2010 data (e.g., [12]) shows a maximum accuracy of around 0.85, which suggests that the current task, while evidently non-trivial, is indeed less complex than the task presented by standard datasets. This hypothesis could be tested by adding more relation types, which is left for future work due to the additional annotation effort required.

4) *What is a good baseline for this task?*

The baseline defined in Section VI shows that the task cannot be fully solved by a simple pattern matching approach. Extending the pattern could improve baseline accuracy, however the boundary between an extended pattern and a full rule-based classifier is not trivially clear.

5) *How do standard machine learning algorithms behave for a small amount of data?*

Both the LSTM model and the SVM model perform well on the relation extraction task. The LSTM model outperforms the SVM model, however this may result from the dimension reduction approach as applied in the design of the SVM.

a) *Which architecture can be used for different machine learning methods?*

For Long Short Term Memory networks, a separate network layer for each feature type and a pooling layer to combine the output of all feature layers is shown to result in high accuracy scores, consistent with results from previous research [12]. For a Support Vector Machine, using a dependency path similarity kernel results in sufficiently high accuracy scores, consistent with [10]. However, the LSTM model outperforms the SVM approach. Note that accuracy (i.e., number of examples correctly classified relative to the total number of examples) is used as an evaluation measure given that the classification task is binary and the classes in the test set are perfectly balanced. In future work the method could be evaluated against all possible token pairs in a given test document. In this case the test examples would be highly imbalanced towards the negative class, which means that precision and recall should be used as evaluation measures instead of accuracy.

The results of the current research provide a first step in the application of information extraction in the law enforcement domain. Subsequent steps include annotation and classification of additional relations (e.g., payment information or sending and receiving goods) as well as semantically more complex processing, e.g., the detection of a false identity from text, i.e., whether the residency relation as extracted from the crime report should be considered real. In future work these further steps need to be investigated in more detail.

ACKNOWLEDGMENT

This research is part of the project *Intelligence Amplification for Cybercrime* which has been funded by the Dutch National Police Innovation Programme.

REFERENCES

[1] R. Schank, "Computer understanding of natural language," *Behavior Research Methods & Instrumentation*, vol. 10, no. 2, pp. 132–138, 1978.

[2] S. Lytinen and A. Gershman, "ATRANS: Automatic processing of money transfer messages," in *Proceedings of AAAI-86*, 1986, pp. 1089–1093.

[3] R. Grishman and B. Sundheim, "Message understanding conference-6: A brief history," in *COLING 1996: The 16th International Conference on Computational Linguistics*, vol. 1, 1996.

[4] G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel, "The Automatic Content Extraction (ACE) program-tasks, data, and evaluation," in *Proceedings of LREC 2004*, 2004.

[5] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. Ó Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz, "SemEval-2010 Task 8: Multi-way classification of semantic relations between pairs of nominals," in *Proceedings of the NAACL HLT Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, 2009, pp. 94–99.

[6] D. Roth and W.-t. Yih, "A linear programming formulation for global inference in natural language tasks," in *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL*, 2004.

[7] S. Ray and M. Craven, "Representing sentence structure in hidden markov models for information extraction," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001, pp. 1273–1279.

[8] N. Kambhatla, "Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations," in *Proceedings of the ACL 2004 Interactive Poster and Demonstration Sessions*, 2004.

[9] A. Culotta and J. Sorensen, "Dependency tree kernels for relation extraction," in *Proceedings of the 42nd annual meeting of the Association for Computational Linguistics*, 2004.

[10] R. Bunescu and R. Mooney, "A shortest path dependency kernel for relation extraction," in *Proceedings of HLT/EMNLP 2005*, 2005, pp. 724–731.

[11] Y. Liu, F. Wei, S. Li, H. Ji, M. Zhou, and H. Wang, "A dependency-based neural network for relation classification," in *Proceedings of ACL/IJCNLP 2015*, 2015, pp. 285–290.

[12] Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng, and Z. Jin, "Classifying relations via long short term memory networks along shortest dependency paths," in *Proceedings of EMNLP 2015*, 2015, pp. 1785–1794.

[13] M. Miwa and M. Bansal, "End-to-end relation extraction using LSTMs on sequences and tree structures," in *Proceedings of ACL 2016*, 2016, pp. 1105–1116.

[14] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W.-t. Yih, "Cross-sentence n-ary relation extraction with Graph LSTMs," in *Proceedings of ACL 2017*, 2017, pp. 101–115.

[15] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao, "Relation classification via convolutional deep neural network," in *Proceedings of COLING 2014*, 2014, pp. 2335–2344.

[16] H. Adel, B. Roth, and H. Schütze, "Comparing convolutional neural networks to traditional models for slot filling," in *Proceedings of NAACL-HLT 2016*, 2016, pp. 828–838.

[17] H.-W. Ng, V. D. Nguyen, V. Vonikakis, and S. Winkler, "Deep learning for emotion recognition on small datasets using transfer learning," in *Proceedings of ICMI '15*, 2015, pp. 443–449.

[18] R. Balabin and E. Lomakina, "Support vector machine regression (LS-SVM) – an alternative to artificial neural networks (ANNs) for the analysis of quantum chemistry data?" *Physical Chemistry Chemical Physics*, vol. 13, pp. 11 710–11 718, 2011.

[19] N. UzZaman, H. Llorens, L. Derczynski, J. Allen, M. Verhagen, and J. Pustejovsky, "SemEval-2013 Task 1: TempEval-3: Evaluating time expressions, events, and temporal relations," in *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, vol. 2, 2013, pp. 1–9.

[20] F. Bex, J. Peters, and B. Testerink, "A.I. for online criminal complaints: from natural dialogues to structured scenarios," in *Proceedings of the ECAI 2016 workshop on Artificial Intelligence for Justice*, 2016, pp. 22–29.

[21] A. van den Bosch, B. Busser, S. Canisius, and W. Daelemans, "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, 2007, pp. 99–114.

[22] CGN Consortium, "Spoken Dutch Corpus," 2003.

[23] S. Tulkens, C. Emmery, and W. Daelemans, "Evaluating unsupervised Dutch word embeddings as a linguistic resource," in *Proceedings of LREC 2016*, 2016, pp. 4130–4136.

[24] S. Lai, K. Liu, L. Xu, and J. Zhao, "How to generate a good word embedding?" *IEEE Intelligent Systems*, vol. 31, no. 6, pp. 5–14, 2016.

[25] H. Zamani and W. B. Croft, "Relevance-based word embedding," in *Proceedings of SIGIR '17*. ACM, 2017, pp. 505–514.