Extracting Complex Relations from Banking Documents

Berke Oral[1,2], Erdem Emekligil[1], Seçil Arslan[1], and Gülşen Eryiğit[2]

[1]R&D and Special Projects, Yapı Kredi Technology
[2]Department of Computer Engineering, Istanbul Technical University
{*erdem.emekligil, secil.arslan*}*@ykteknoloji.com.tr*
{*oralbe, gulsen.cebiroglu*}*@itu.edu.tr*

Abstract

In order to automate banking processes (e.g. payments, money transfers, foreign trade), we need to extract banking transactions from different types of mediums such as faxes, e-mails, and scanners. Banking orders may be considered as complex documents since they contain quite complex relations compared to traditional datasets used in relation extraction research. In this paper, we present our method to extract intersentential, nested and complex relations from banking orders, and introduce a relation extraction method based on maximal clique factorization technique. We demonstrate 11% error reduction over previous methods.

1 Introduction

Despite recent efforts for digitalization in banking domain, formal letters (such as orders, petitions, demands or complaints) still remain as one of the main communication media in corporate banking. A mid-to-large scale bank receives millions of orders in a year, most of which are money transfer requests. Reading and manually processing those documents require significant amount of human labour. Moreover, since these documents require specialized knowledge to be interpreted, employment and education of trustable personnel is often difficult. This situation makes the automatization of the process crucial.

Automatic processing of banking documents is an Information Extraction (IE) task, where one seeks to extract structured information from semi-structured or unstructured texts. This is usually conducted with pipelined processes. The one used in this study depicted in Figure 1. The first step is to extract entities of interest from raw text, in our case is extracted via Optical Character Recognition (OCR) system. This can be done with Named Entity Recognition (NER) algorithms or pattern

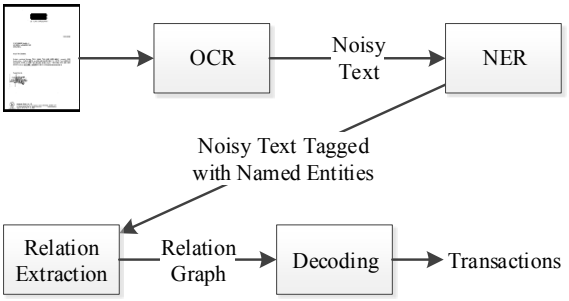

Figure 1: Information extraction pipeline.

based methods. Secondly, semantic relations between extracted entities are predicted with Relation Extraction (RE) algorithms. Finally, semantic structures are constructed on top of the relations as higher order structures such as templates or events.

NER is an important step for many Natural Language Processing (NLP) tasks. It is a widely researched area, and current state-of-the-art algorithms (Lample et al., 2016) give satisfactory results in most situations. RE is considered as a linguistically higher order task compared to NER, and proposed algorithms are often tailored to problems in hand. Majority of research in this area is focused on extracting binary relations. Although there are some research dealing with complex relations (McDonald et al., 2005; Peng et al., 2017), work on nested complex relations as in the case of banking transactions is scarce.

In this paper, we focus on money transfer requests as the major process type in banking orders. Within these documents, we seek to extract *transactions* as our structured information of interest. Each order may contain one or more transactions, which are made of sender and receiver account numbers, names, bank information as well as details of transaction process such as the transfer amount and its currency type. In order for a transaction to be valid, its sender, receiver and pro-

Proceedings of the Second Workshop on Economics and Natural Language Processing, pages 1–9
Hong Kong, November 4.

cess details should be clearly defined. Thus, in our system, we abstract these real-world entities as three main *divisions* of a transaction: sender, receiver, and process details. Each of these divisions contains required (emphasized with bold font in Figure 2) or optional *slots*. Sender and Receiver divisions must either include an Account Number or IBAN (having either one of these is enough to address the account). Process Details must contain transfer Amount and its Currency.

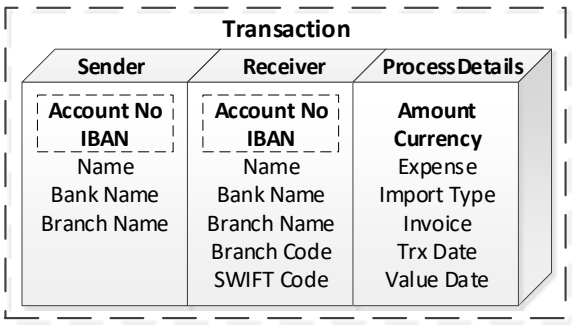

Figure 2: Divisions of a transaction and their slots.

Slots of the divisions should be filled with named entities extracted from a document. Named entities related to the same slot might have been stated multiple times by the author in different places of the document. For example, the name of the account holder may occur within both the body text and the signature part. Additionally, a single named entity may carry information for multiple slots belonging to different transactions. For example, multiple money transfer orders might have been given from a single sender account, hence an entity holding the sender IBAN should be linked to different transaction slots. Figure 3 provides such a sample document with two transactions. Private information were masked on the figure, and extracted entities were specified with colored boxes.

In this paper, we focus on the relation extraction and the decoding stages of the pipeline introduced in Figure 1.[1] We propose a method that can automatically extract transaction information from banking documents by forming nested complex relations using a relation graph. Our algorithm first predicts binary relations between entities. This forms an undirected graph, where nodes are entities, and edges are predictions. On this graph, the algorithm performs series of maximal

[1] For the OCR stage of the pipeline, we use Abbyy Finereader v12 system. For the NER stage, we use an adaptation of Lample et al. (2016) for banking documents.

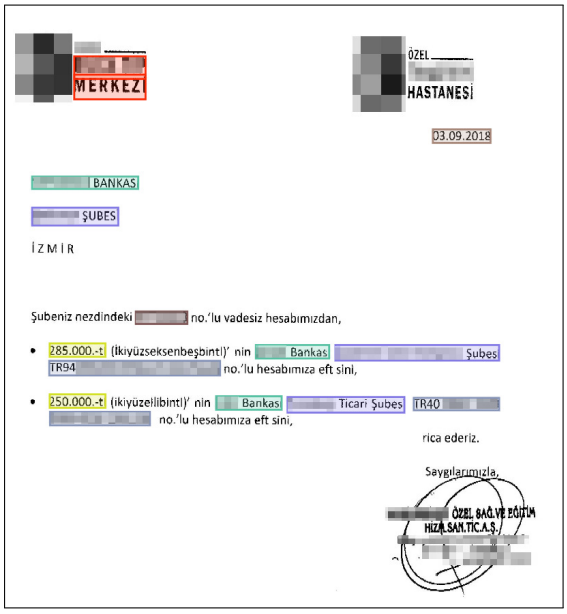

Figure 3: A sample money transfer order document with two different transactions provided in separate lines starting with bullets. Best viewed in color.

clique factorization operations to form transactions.

In Section 2, we discuss the previous work on intersentential complex relations. In Section 3, we describe a transaction extraction method that uses maximal cliques in a predicted relation graph to extract complex relations. Lastly, we discuss comparable approaches to our method in Section 4, and give our remarks in Section 5.

2 Related Work

In early IE systems (Chinchor, 1998), extracting complex relations (i.e database entries, templates) was mostly accomplished with rule based approaches. To the best of our knowledge Chieu and Ng (2002) was first to extract complex relations from binary relations using maximal clique approach. In biomedical domain, McDonald et al. (2005) extended this approach by adding predicted probabilities from a trained classifier. Using geometric mean of the relation probabilities in cliques, they selected highest scoring cliques as complex relations. Rather than using maximal cliques, Wick et al. (2006) used a clustering algorithm to construct complex relations. They trained a classifier that computes similarity score between two clusters. Starting from singletons, their clustering algorithm built relation tuples.

Event extraction also deals with complex relations. In this task, algorithms often detect a trigger

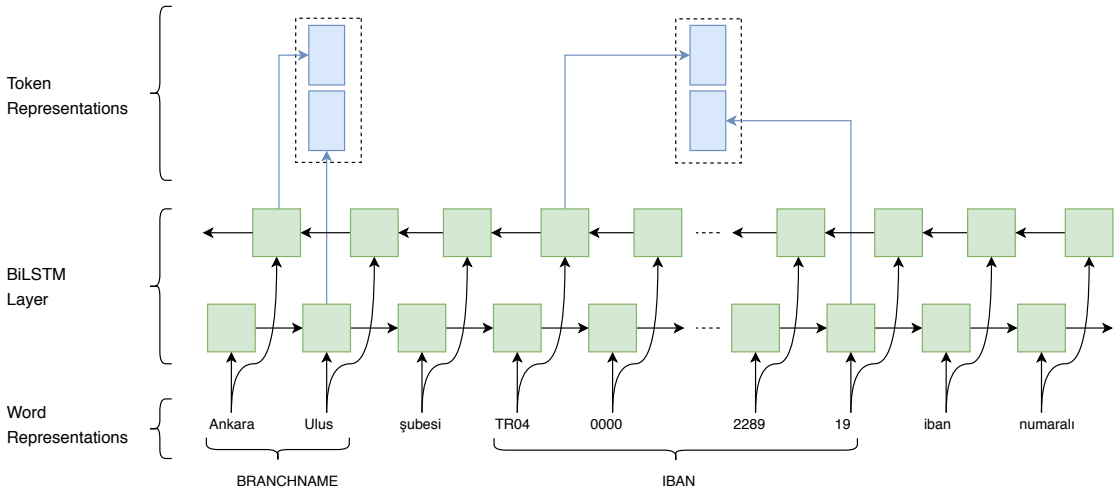

Figure 4: Entity representations

word for an event, then using the trigger word and its relations, arguments of an event are filled. Our approach to transaction extraction is analogous to event extraction. In our case, the trigger is a group of amount entities. An interesting approach in this area is the use of dependency parsing algorithms as means of complex relation extraction (McClosky et al., 2011; Sprugnoli and Tonelli, 2017; Wang et al., 2018). In literature, closest work to ours is by Şahin et al. (2018), which uses a non-projective transition based parser to extract transactions from banking documents.

Although most of the research in relation extraction literature focused on binary relations from single sentences, n-ary and document level relation extraction has become increasingly popular in recent years. Notably, Peng et al. (2017) proposes a graph LSTM architecture, which runs on two directed acyclic graphs constructed from syntactic dependency trees of sentences. In these graphs, syntactic roots of consequent sentences are linked together. This allows the algorithm to capture syntactic features between entities in proximate sentences. Later, their model predicts n-ary relations among fixed number of entities. Jia et al. (2019) create entity representations from different discourse sizes (document, paragraph, sentence) and predicts relations for each entity tuple in a document. In this approach, as the number of entities in document increases, possible n-ary relations will explode (2^n), which makes this approach for n-ary relation extraction computationally not feasible. However, they merge multiple mentions of a same entity (e.g same gene mentioned in different paragraphs or sentences) into one representation, which reduces the number of entities in the relation extraction step. They argue that this step makes n-ary relation extraction computationally affordable. In our case, the number of required slots, the complexity of relation types, and the existence of multiple mentions referring to the same transaction slots (occurring under very distinct surface forms also due to OCR errors) make the task even more challenging.

3 Method

We propose a method to extract transactions from banking documents, given a sequence of words and their named entity types.[2] It first creates a representation for each entity within the text, then predicts an undirected relation for each entity pair. This creates a fully-connected graph that our decoding algorithm uses to construct transactions.

3.1 Relation Extraction

In this section, we describe the architecture and the training details of our relation extraction model. Each banking document is provided as a single sequence to our model, which predicts a relation type for each entity pair. We use a BiLSTM to create contextual representations for our entities and two entity representations are concatenated then fed into a multilayer perceptron (MLP) with three hidden layers to predict relations. This creates an undirected graph which is represented by matrix $R^{N \times N}$ where N is the number of entities in a graph. We calculate the cross entropy loss for

[2]Our named entity types are named as the same with slot names shown in Figure 2.

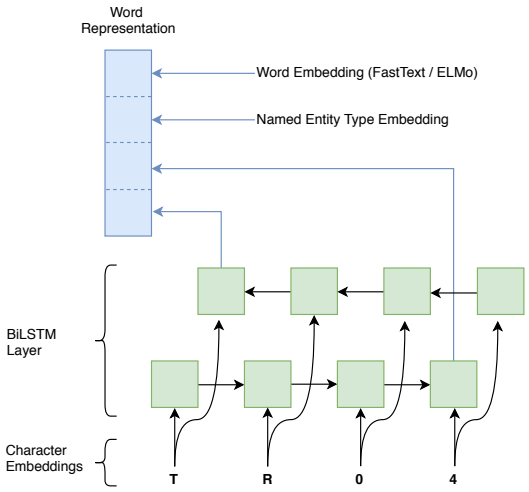

Figure 5: Word representations

each element of the matrix and take their sum as document loss.

In banking documents, entities are often formed of multiple words (e.g., branch name, IBAN entities in Figure 4). Therefore, we need to derive the entity representations from multiple word representations. As the overall entity representations, we use the concatenation of forward and backward LSTM outputs at the entity boundaries (right boundary token for the forward pass and left boundary token for the backward pass). Figure 4 visualizes our entity representation. Two dotted rectangles on the top represent the embeddings for the two entities branch name and IBAN composed of 2 and more than 4 tokens respectively. The arrows coming to the dotted rectangles depict the representations extracted from the entity boundaries.

Each word in our model is reprensented as the concatenation of three vectors: pretrained word embeddings, character level word representations, and named entity type embeddings (Figure 5). We pre-trained our word-embeddings via both FastText (Bojanowski et al., 2016) and ELMo (Peters et al., 2018) by using a corpus of 626M words collected from banking domain. Using the standard parameters, both models were trained for 10 epochs. FastText embeddings were loaded into a lookup table. To represent out of vocabulary (OOV) words in FastText embeddings we used zero vector with same of the embedding dimension. Our documents contain plenty of numerical values crucial for our task such as account numbers, amounts, dates etc. Those appear diverse surface forms, yielding to rare occurrence counts. We used a word transformation algorithm in order to represent them with FastText. The algorithm is used if the number of letters in a word is less than or equal to the half of the word length. It replaces the word with a token specifying the count of letters (L), digits (D) and punctuation characters (O) it contains (see Table 1 for examples). Because ELMo creates word embeddings from characters, we did not need to apply this transformation for ELMo representations. Character level representation of words are also created again by a BiLSTM layer similar to Lample et al. (2016). Both character embeddings and named entity type embeddings are initialized randomly and learned during training.

Word	Token
22/05/2019	<L0D8O2>
TR04	<L2D2O0>
10,000	<L0D5O1>

Table 1: Examples of word transformation algorithm

We implemented our models using Tensorflow framework. Number of words in documents and number of characters in words vary for each instance. We used mini-batching during training. Each word and character padded to longest element in the mini-batch with padding tokens so that each instance has the same sized vector. Similarly, since the number of entities in the documents also varies, we padded the relation matrices in the mini-batches with padding entities and NONE relations. During training, we did not use the padded cells in loss calculation. For optimization we used Adam (Kingma and Ba, 2015) algorithm with learnin rate of 1e-3. We used annealing of 0.9 after every epoch. For regularization, we used dropout (0.5) at the BiLSTM inputs and before each hidden layer in MLP.

3.2 Relation Structure

In order to correctly extract the transactions within a money transfer order document, one needs to correctly determine the transaction count and fill their slots (Figure 2). This kind of complex structures may be formulated as binary relations (McDonald et al., 2005). We defined an undirected relation structure unique to solve money transfer orders where multiple transactions may occur within a single document and some entity types (e.g., name, bankname) are shared between sender and receiver divisions of a transaction. We defined 5

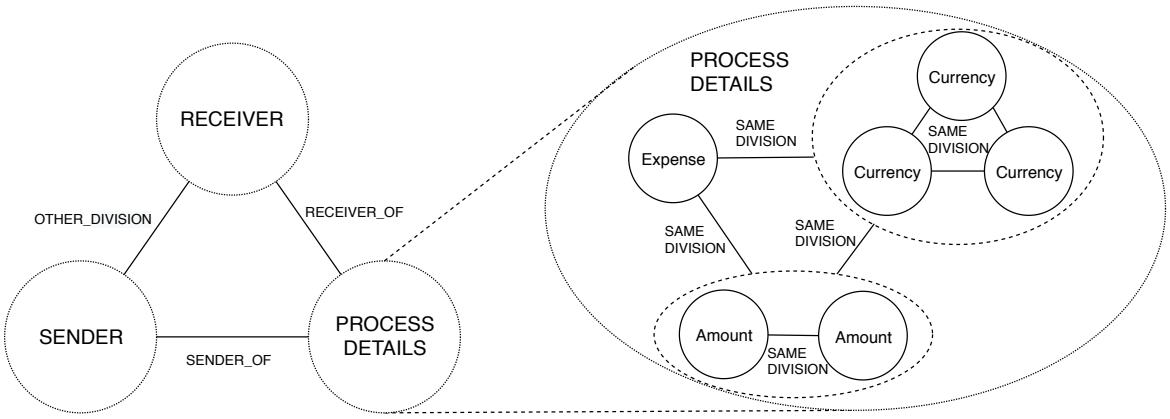

Figure 6: An example relational graph containing a transaction.

different relation labels (detailed in Table 2) in order to distinguish the divisions to which the entities belong within a transaction. For instance, entities that are connected with RECEIVER_OF relation are either in receiver or process details divisions. Since possible entity types that reside in sender and process details divisions do not intersect (Figure 2), we can distinguish entities of these two divisions although the relations are undirected. Below, we describe our algorithm to decode multiple transactions within an order. In Figure 6, we provide a visualization for the relations within a single transaction. For simplicity, we provide only the abstract divisions on the left side of the figure and the relations between them meaning that entities belonging to these divisions will be related to the entities of the other division with the mentioned relation types. On the right side of the figure, we zoom into one of the divisions (process details) where entities belonging to a same division are connected to each other with SAME_DIVISION relation type. As explained in the previous sections, named entities occurring within the documents might have been stated multiple times by the author in different places of the document. The figure depicts (with dashed ellipses) a scenario where the same currency is repeated (maybe with slight surface form differences) 3 times within the order and the amount is repeated 2 times. One should keep in mind that this is just a visualization and we actually keep this fully-connected undirected graph as a binary relation matrix $R^{N \times N}$.

Relation Type	Description
NONE	Entities are not in the same transaction
SAME_DIVISION	Entities are in the same division
OTHER_DIVISION	One entity is in Sender while other is in the Receiver division
RECEIVER_OF	One entity is in Receiver while other is in the Process Details division
SENDER_OF	One entity is in Sender while other is in the Process Details division

Table 2: Possible relation types and their description

3.3 Extracting Transactions

The first step of our decoding algorithm is to determine the number of possible transactions. In order to achieve this, the decoding algorithm creates a sub-graph from amount entities. Since the entities of different transactions are expected to be connected with NONE relation type, each maximal clique of this sub-graph can be said to belong to different transactions.[3] However mistakes in relation extraction process complicate this. Missing or extra edges between nodes (i.e., entities in our case) can result in extra cliques, where many of their nodes are shared. Algorithm uses cardinality of maximal cliques to discard possibly wrong ones. It compares cliques to each other and eliminates the smaller clique if it has a node that intersects with larger clique. If two cliques has same the cardinality while sharing a node, it randomly eliminates one of the cliques. The result of this process is an unconnected set of cliques which are treated as the roots of separate transactions.

Once the transaction root nodes are discovered, i.e the amount slots (the process details division) for each transactions are determined, it becomes

[3] Although some slots may be shared between transactions, it is a convention that amounts are specified explicitly for each transaction. In fact, in our data there exist no amount entities that are shared between transactions.

possible to fill the other slots by using the relation graph. In a perfect world, where all relations are extracted correctly, this slot filling is very trivial: each entity value will be written to a slot regarding to the relation type of that entity to the root node. Due to the errors propagating from the relation extraction stage, we need a decoding algorithm to fill out the missing parts of the transactions. To fill each slot, our algorithm first looks up all the entities that has matching named entity types and creates a sub-graph from these if they are connected to the amount. For example, if we aim to fill the sender division's bank name slot for a transaction, we will extract all bank name entities connected with SENDER_OF relation to the amount entities of that transaction. This time, we select a maximum clique of the entities connected with a SAME_DIVISION relation and fill the slots with their values.

Only transactions that contains required slots (sender and receiver division's account number or IBAN slots, process details division's amount, and currency slots) are considered as valid transactions. In the last step of our decoding algorithm, we pruned the outputs transactions that is not containing those required slots.

With the above decoding algorithm, it is possible that re-occuring entities (with exact or different surface forms) are assigned to a single slot. One may use a post-processing stage to select the best entity value for a slot.[4]

4 Experimental Results

4.1 Dataset

Our dataset contains 3500 Turkish banking documents with a total of 4102 transactions. In order to represent different types of layouts, the dataset is collected such that each document is from a different customer.[5] The dataset contains 51,396 entities and 1.17 transactions per document, 6.7% percent of them contains multiple transactions. On average, there are between 1.18 to 2.23 entities per slot depending on the slot type.

The documents are in image format since they are received via fax, scanner or email channels. They contain misrecognized characters, extra or missing characters or spaces, incorrect token sequences and so on due to noisy images and OCR errors. In our experiments, we randomly selected 600 of these documents (containing 730 transactions in total) as our test set and 400 of them as the validation set.

4.2 Experiments and Discussions

Our algorithm predicts a relation type for each entity pair in a document. This creates a relation graph which is represented as an $N \times N$ matrix where N is the number of entities. To evaluate relation extraction performance, we used F1 measure. Since the relations are undirected, we used only the upper triangles of relation matrices in our evaluations. Since the cells on the diagonal line of the relation matrix are always expected to contain SAME_DIVISION relations, they are excluded from the evaluations. Table 3 gives relation extraction scores of our model both with FastText and ELMo embeddings. The dataset contains many rarely occurring words due to noisy OCR outputs, especially in numerical tokens and proper nouns. We observe that such rare words are handled better by ELMo.

Relation Label	Count	Model w/ FastText	Model w/ ELMo
NONE	16786	74.18	84.97
SAME_DIVISION	17948	88.56	93.10
OTHER_DIVISION	13000	97.08	95.95
RECEIVER_OF	13333	90.26	92.43
SENDER_OF	15156	97.27	96.49
Macro Avg		89.47	92.59
Micro Avg		88.88	92.35

Table 3: F1 scores of binary relation extraction step

In order to measure the performance of our transaction extraction stage, we used a slot level entity matching evaluation. Entities occurring in both predicted and gold slots were counted as true positives, the remaining entities within a predicted slot were counted as false positives, and the missed entities of a gold slot were counted as false negatives. Since each page may contain multiple transactions, while evaluating the predicted transactions, we first needed to match them with gold ones. We computed a similarity score for each predicted-gold transaction pair in a document and selected highest scoring pairs. The similar-

[4]The post-processing is crucial for our task since the input source of our data is OCR and the data is noisy. For example, we use an IBAN validator at this stage.

[5]Customers would usually prefer to use their own document layout (templates) for their consecutive transaction orders yielding documents with similar layouts.

		Rule Based	Trans. Based	Model w/ FastText	Model w/ ELMo
Sender	P	80.32	87.21	94.61	**96.05**
	R	60.19	72.42	81.13	**82.65**
	F1	68.60	79.10	87.34	**88.79**
Receiver	P	78.49	87.07	91.99	**92.45**
	R	56.33	69.00	81.74	**83.39**
	F1	65.29	76.91	86.47	**87.58**
Details	P	87.56	91.39	94.63	**95.33**
	R	56.14	69.21	85.35	**87.25**
	F1	68.00	78.56	89.70	**91.04**
Overall	P	82.86	89.21	93.86	**94.59**
	R	57.32	70.18	83.35	**85.10**
	F1	67.35	78.41	88.22	**89.50**

Table 4: Slot level entity matching macro average scores grouped by divisions.

ity score was geometric mean of entity matching scores (F1) between required slots (sender's and receiver's account numbers and IBAN, amount, and currency of transaction). We set a rule where predicted transactions could not match with gold transactions if the similarity score was 0, or they were already matched with other predicted transactions. Unmatched predicted transactions were counted as *wrong*, and all of their entities were counted as false positives during slot level evaluation. Similarly unmatched gold transactions were counted as *missed*, and their entities were counted as false negatives. Table 4 provides slot level entity matching scores grouped by divisions and as overall.

We compare our method with two models: the rule based and the dependency based approaches from Şahin et al. (2018). The rule based method is derived from banking conventions and some basic patterns. It chooses the first seen account entity as sender and remaining ones as receiver. Other entities are set using a similar logic that also considers their proximity to divisions. For instance, it can select the closest currency entity to the amount as its currency.

We also adapted Şahin et al. (2018)'s dependency parsing method to our data, which resulted in increased number of entity and relation types. Since, our dataset contains more entity types (14 vs 7 in Şahin et al. (2018)), during the adaptation, we needed to add more relation types to the ones in the original study. This approach uses a transition based non-projective dependency parsing model (Nivre et al., 2009) to attach account numbers to the root of a dependency tree as Sender/Receiver, and related entities (name, bank name, etc.) to Sender/Receiver account numbers. Amounts are attached to receivers while currency and other process details are attached to amounts. This approach makes an assumption that each document can only have one sender. Since, each receiver carries all the unique entities of a transaction, multiple transactions can easily be constructed.

Rule-based method cannot relate more than one entity with a slot. As a result, it performs marginally worse in recall. To correctly fill a slot, it is enough to find one true entity in the document. However, since we are working on a text coming from an OCR system, entities are often misspelled. Having more than one entity in a slot is advantageous for post-processing steps.

Our model clearly outperforms both the rule-based and the dependency parsing methods. The use of ELMo embeddings gives consistent improvements in both precision and recall. Although the dependency parsing model achieves satisfactory results in precision, its performance in recall is poor. We argue that this is due to the way the dependency trees are constructed in Şahin et al. (2018). Each dependent may be attached to one head, therefore in slots that have multiple true entities (e.g in cases where sender's name stated multiple times), entities are expected to be attached to each other with a directed dependency relation (called "*SAME*") according to their occurrence order. Then, the first entity within this order is expected to be attached to its true head. According to our view, this is kind of a unnecessarily strict grammar for the semantic problem in hand. In our investigations over the predicted outputs, we see that the transition based parser struggle to detect correct order for *SAME* relations, thus degrading its parsing accuracy.

Figure 7 shows the average slot level score difference between our model (FastText) and transition-based dependency parsing model. In slots where there are multiple entities, the difference in recall is consistently higher than the cases where we have one entity for a slot. It is also interesting that precision scores of the two methods are very similar. However, in empty slots, dependency parsing model has much higher number of false positives: 702 in the dependency parsing model vs. 304 in the FastText and 192 in the ELMo models. We interpret this as LSTMs better ability to

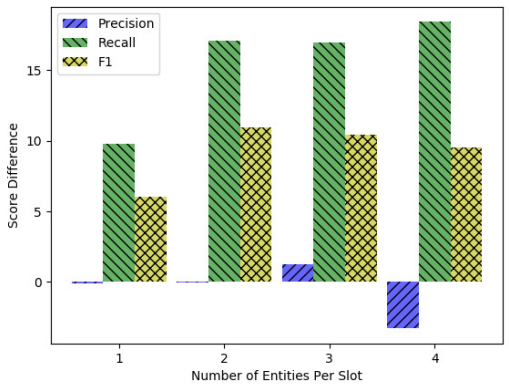

Figure 7: Score difference between our model (w/ FastText embeddings) and transition based parser model. Horizontal axis shows number of possible entities for a slot.

# Errors	Rule Based	Trans. Based	Model w/ Fasttext	Model w/ ELMo
0	74	372	454	490
1	52	43	47	40
2	81	70	52	39
3	62	17	21	13
4	80	4	11	7
5	63	5	9	4
6+	91	12	4	13
# Matched	503	539	601	606
# Missed	227	191	129	124
# Wrong	67	64	27	13

Table 5: Number of entity matching errors in predicted&matched transactions. Number of *matched*, *missed*, and *wrong* transactions are also given in bottom rows.

learn semantics of pages.

We also counted the number of entity matching errors in order to evaluate the quality of predicted and matched transactions. Table 5 provides these statistics (in the upper block) as well as the numbers of missed and wrong transactions of each model (in the lower block). Unlike in the slot level evaluation, performance of the rule based model is not dramatically worse than the other methods based on the matched transaction counts, but the predicted transactions have lower quality. Rule based model could predict only 14.7% of the transactions with no error in any of their slots, while this ratio is much higher in other models. Similarly to our slot level evaluations, our models give better performance than the other two methods, while the use of ELMo embeddings provides consistent improvements.

5 Conclusion

In this paper, we introduced a method to extract transactions from banking documents. The method uses BiLSTM based deep neural network to predict relations between each entity pairs, and creates a relation graph. From this graph, our decoding algorithm constructs a series of sub-graphs and applies maximal clique factorization to determine number of transactions and fill their slots. We demonstrated that our method is more accurate at predicting transactions and their slots than previously proposed methods. Moreover, It has a higher recall rate on slots with multiple entity candidates. This allows the use of excess entities in post-processing steps, which can mitigate the mistakes of OCR system.

Acknowledgments

This work is supported by The Scientific and Technological Research Council of Turkey with the project no TEYDEB 3180571. We would like to thank our colleagues Mehmet Yasin Akpınar, Cemil Cengiz, Deniz Engin, and Tuğba Pamay for their valuable discussions and support.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Hai Leong Chieu and Hwee Tou Ng. 2002. A maximum entropy approach to information extraction from semi-structured and free text. In *Eighteenth National Conference on Artificial Intelligence*, pages 786–791, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Nancy A. Chinchor. 1998. Overview of MUC-7. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*.

Robin Jia, Cliff Wong, and Hoifung Poon. 2019. Document-level n-ary relation extraction with multiscale representation learning. *CoRR*, abs/1904.02347.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1626–1635, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Seth Kulick, Scott Winters, Yang Jin, and Pete White. 2005. Simple algorithms for complex relation extraction with applications to biomedical IE. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 491–498, Ann Arbor, Michigan. Association for Computational Linguistics.

Joakim Nivre, Marco Kuhlmann, and Johan Hall. 2009. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 73–76, Paris, France. Association for Computational Linguistics.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

R. Sprugnoli and S. Tonelli. 2017. One, no one and one hundred thousand events: Defining and processing events in an inter-disciplinary perspective. *Natural Language Engineering*, 23(4):485–506.

Shaolei Wang, Yue Zhang, Wanxiang Che, and Ting Liu. 2018. Joint extraction of entities and relations based on a novel graph scheme. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 4461–4467. AAAI Press.

Michael Wick, Aron Culotta, and Andrew McCallum. 2006. Learning field compatibilities to extract database records from unstructured text. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 603–611, Sydney, Australia. Association for Computational Linguistics.

Gözde Gül Şahin, Erdem Emekligil, Secil Arslan, Onur Ağın, and Gülşen Eryiğit. 2018. Relation extraction via one-shot dependency parsing on intersentential, higher-order, and nested relations. *Turkish Journal of Electrical Engineering & Computer Sciences*, 26(2):830–843.