RELATION EXTRACTION IN TURKISH

by

Murat Buldu

Submitted to the Department of Computer
Engineering in partial fulfillment of
the requirements for the degree of
Bachelor of Science

Undergraduate Program in Computer Engineering
Boğaziçi University

Fall 2019

RELATION EXTRACTION IN TURKISH

APPROVED BY:

Tunga Güngör .....
(Project Supervisor)

DATE OF APPROVAL: DD.MM.YYYY

ABSTRACT

RELATION EXTRACTION IN TURKISH

Information extraction from texts is an interesting problem and there are lots of work on it. With the introduction of the word embedding, the methods used in relation extraction have changed from the methods based on hand crafted features to the methods whose features are extracted from neural networks. The previous works about relation extraction in Turkish use the methods with hand crafted features and the number of work is not much. In this paper, we focus on the methods with neural networks in Turkish texts and compare the results of the models and the effects of the each component.

ÖZET

TÜRKÇE'DE İLİŞKİ BULMA

Yazılardan bilgi çıkarma ilginç bir problemdir ve üzerinde pek çok çalışma yapılmıştır. Word embeddinglerin kullanılmaya başlanmasıyla, kelimeler arasında ilişki bulmak için kullanılan yöntemler, elle oluşturulmuş özelliklere dayalı yöntemlerden, sinir ağlarıyla oluşturulmuş özelliklere dayalı yöntemlere kaydı. Türkçede ilişki bulma konusunda önceki çalışmaların çoğu elle oluşturulmuş özelliklere dayalı yöntemlerden oluşuyordu ve bu konuda çok fazla bir çalışma da yoktu. Bu çalışmada sinir ağlarıyla oluşturulmuş özelliklere dayalı yöntemleri Türkçede yazılarda deneyip, bu modellerin etkilerini inceleyeceğiz.

TABLE OF CONTENTS

1. INTRODUCTION AND MOTIVATION

The problem of relation extraction can be described as finding the relation between the given two nominal words in a sentence. Also, there are some works with end to end approach, and so the nominal words are also predicted, in these models.

Relation extraction is an important task in information extraction in natural language processing. It can be used in different domains. Some of them are question answering, medicine and drug discovery.

Because of these applicability, relation extraction task also attracts the attention of the researchers and there are lots of work on this subject. To compare these methods, there are some challenge datasets such as SemEval datasets but these datasets contains English texts. So the number of work done in other languages is much lesser. So we want to apply some of these methods on Turkish texts.

The first attempts to the relation extraction task focused on the hand crafted features and extracting these features required domain specific knowledge. As a result, these methods were poor in generalization of the general task. The more recent works mainly used word embeddings and features extracted from deep neural networks. One of the main advantage is that these methods does not require domain specific knowledge and can be generalized more easily. These methods will covered in a more detailed way in the following parts.

In the following parts, we will discuss some related work in Literature Review, the applied methods in Methods, the results could be obtained by our experiments in Results, a general discussion about the results of our methods in Conclusion and Discussion and what can be next in Future Work.

2. LITERATURE REVIEW

2.1. Reviewed Papers

The paper [1] is Relation Classification via Convolutional Deep Neural Network, which uses convolutional layer for feature extraction.

The relation classification problem is described as "given a sentence and given two words from this sentence, find the relation between the given words" in the paper. Therefore, the nominal words should be specified to the system.

The idea of using convolutional neural networks was mentioned by Collobert et al. (2011) for POS tagging, chunking, Named Entity Recongnition and Semantic Role Labeling. This paper uses a similar idea for relation classification and uses a convolutional layer to extract features.

The neural network architecture consists of 3 layers: Word Representation, Feature Extraction and Output. There is no additional preprocessing operations in the system, so the input of the system is a sentence with two specified words. Then, each word token is transformed into real valued vectors using word embeddings. Then, the lexical and sentence level features are extracted and fed into the softmax classier. The dimension of the output is equal to the number of relation types and the output values correspond to the confidence scores of the relations.

In the word representations, the words in the sentence are turned into vectors using word embeddings. These word embeddings are learned from lots of unlabeled data and they need so much time for training. There are available pretrained word embeddings, and in this paper, the word embeddings provided by Turian et al.(2010) are used.

The features of the system can be categorized into two groups, lexical level fea-

tures and sentence level features. There are five lexical level features. L1 is the word representation of the first nominal word. L2 is the word representation of the second nominal word. L3 is the word representations of the left and right tokens of the first nominal word. L4 is the word representations of the left and right tokens of the second nominal word. L5 is the word representations of the hypernyms of the nominal words from WordNet. To extract sentence level features, word features and position features are fed to a max pooled convolutional neural network. For word features, the windows of the word representations are used and in this paper, windows size is set to 3. The position features are the combination of the relative distances of the current word to nominal words. Then, these relative distances are mapped to a vector of dimension d_e. These word features and position features are concatenated and fed to a max pooled convolution layer. Then, a fully connected layer with *tanh* activation function is applied to get sentence level features.

In the output layer, lexical level features and sentence level features are concatenated and fed into a softmax classifier, and each output of the softmax layer is the confidence score of the corresponding relation.

The log likelihood of the parameters are used as cost function, and during the training, the cost function is minimized with stochastic gradient descent.

In the experiments, SemEval-2010 Task 8 dataset is used. This dataset contains 8000 training and 2717 test instances. There are 9 relationships and an undirected Other class in the dataset.

The effects of lexical level features and position features are examined and the results show that each of the lexical level features and position features increases the performance. The results show that the performance of the system is better than the traditional models.

The next paper [2] is Classifying Relations by Ranking with Convolutional Neural Networks, which also uses convolutional layers to extract features. In this paper, a new

pairwise loss function was proposed. The experimental results show that this approach performs better, omitting Other class increases the performance and if only the text between the two nominal words are used, using only word embeddings as input is sufficient.

In the proposed neural network, given a sentence, CR-CNN computes a score for each relation class. During the training, the system learns the class embedding matrix, which contains the class representation of the relation classes as columns.

The first layer of the proposed neural network is word embeddings. Each word token is transformed into a real valued vector representation. The word embeddings is learned in the system and the size of the vector is chosen by user.

Word position embeddings are also used in the system. Word position of a word is described as the relative distances of the word to the nominal words. Then, this relative distances are mapped to a vector of dimension d^{wpe}. Word position embedding matrix is initialized randomly and the dimension is a hyperparameter of the system. Then, the word embeddings and the word position embeddings of the inputs are concatenated to form the feature vector.

These features are fed to a max pooled convolutional layer to get the sentence representation. Since sentence sizes may be different, using a convolutional layer is a good fit. First convolutional layer computes the local features around each word. Then, local features are combined into a fixed sized vector with a max operation.

The proposed neural network computes the score of each relation class with using these sentence representation and class embedding matrix. The dimension of the class embedding must be equal to the dimension of the sentence representation.

During the training, a pairwise ranking loss function is minimized. There are two terms in the loss function, and the idea of the loss function is that the first term decreases as the score of the correct class increases and the second term decreases as the

score of the wrong class with highest score decreases. Sampling informative negative classes can increase the performance of the learned model. For the tasks with large number of classes, a fixed set of classes could be used as negative classes.

In this paper, an artificial class is described as the class which contains the all examples which does not belong to other classes. "Other" class is an example of an artificial class. These artificial classes are removed from the class embeddings. During the training, then an example of an artificial class enters the system, the first term in the loss function is set to zero and for the prediction, if all the classes have a negative score, the sentence is classified as "Other".

In this paper, SemEval-2010 Task 8 dataset is used. This dataset contains 9 relation classes and an artificial class "Other". For word embeddings, pretraining was performed using word2vec tool on the English Wikipedia corpus, and this pretrained word embedding was used during the experiments.

In the first experiment, the effect of word position and using the text between the nominal words are discussed. The results show that using word positions increases the performance, but if only the text between the nominal words is used, the improvement is so small. In the next experiment, the effect of omitting the artificial class is discussed. The results show that omitting the artificial class improves the performance of the system. In the next experiment, the performances of the CR-CNN and CNN+Softmax are compared. The results show that CR-CNN performs better.

In the paper, the most representative trigrams for each class are extracted. With tracing back the sentence representations, the responsible trigrams could be identified. The most representative trigrams have the biggest impact on the score, so they are a good indicator about how the system classifies the sentences.

The next paper [3] is Bidirectional Long Short-Term Memory Networks for Relation Classification. In the paper, bidirectional long short-term memory networks is proposed to extract the features with complete and sequential information. Long

distance relationships are captured in this networks. Additional lexical resources are also used in the neural network such as WordNet, NLP tools like dependency parser, named entity recognizers. The results show that using additional features improves the performance of the network.

It is mentioned that there are many different neural network models used to classify the relation between the nominal words, in the paper. The main differences are that BLSTM is used to extract the sentence level features, and that additional NLP tools are used.

Bidirectional LSTM consists of one network in forward direction and one network in backward direction. As a result, for every point in the network, the network has complete, sequential information about all points before and after it.

In the proposed network, the features are divided into two groups. The first one is lexical features, which consists of word, POS, NER and hypernyms. The other one is relative position relationship features. There are three relative position relationship features. The first one is position feature. Position feature is the vector obtained by mapping the relative distances of the current word to the nominal words. The second one is relative dependency features. It contains the relative positions of the current word to the root, the first selected word and the second selected word. The last one is dep features, which is the tag of the current word to its parent in the dependency tree. Then, these features are concatenated.

These features are fed to a BLSTM to extract sentence level features. The memory cell outputs and the cell outputs of the both forward and backward directions are concatenated. Then, these outputs are divided to three group according the position to the nominal words. Then, max pooling is applied to the first two group and the last two group, and the results are concatenated to form the sentence level features.

Since the focus of the lexical level features is the nominal words, lexical level features are constructed by concatenating the initial features of the nominal words and

the outputs of the nominal words from the BLSTM. Then, these lexical features and sentence level features are concatenated and fed to a softmax classifier to predict the relation between the nominal words.

The dataset used in this paper is SemEval-2010 task 8 dataset, which includes 8000 training and 2717 test instances. There are 9 relation types and "Other" relation, which contains all instances which are not related with these 9 relations.

During the experiments two pretrained embedding is used, Turian et al. (2010) with embedding size 50, and Jeffrey Pennington et al. (2014) with embedding size 100 to understand the effect on the performance.

In the first experiment, the effect of the additional features was tested. One network trained with only word embedding as input and one trained with all features. The results show that the performance of the network trained with only word embeddings is close to the performance of the CR-CNN trained with only word embeddings, and the performance of the network trained with all features is higher than the performance of the CR-CNN.

In the next experiment, the effect of different features was tested. The results show that removing named entity and position features do not change the performance much, the reason might be the other features contains the information of NER and position features.

In the next experiment, the effect of different sized embeddings was tested. The results show that the performance of the network with embedding size 100 is higher than the performance of the network with embedding size 50. Since the embedding size is higher, word embeddings could contain more information.

In the next experiment, the effect of using only one direction was tested. The results show that BLSTM performed better than unidirectional LSTM. Moreover, doing the max operation without dividing into three groups for the sentence level feature

extraction decreased the performance of the network.

The next paper [4] is End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In this paper, an end to end neural network is proposed to classify the relations in a sentence. This network uses both word sequences and dependency trees by using bidirectional tree structured LSTMs and bidirectional sequential LSTMs. Entity pretraining and scheduled sampling are used to make the nominal word selection better in early stages.

The aim of the proposed network is using word sequences and dependency trees construct an end to end relation extraction system. This neural network first predicts the nominal words, and then finds the relation between the nominal words.

The model contains three layers: a word embedding layer, a word sequence layer and a dependency layer.

In the embedding layer, word tokens are transformed into vectors using word embeddings, part of speech (POS) tags, dependency types and entity labels.

In the sequence layer, the sentences are considered as a linear sequence of words. A bidirectional LSTM is used to extract the sequential features from the sentence. The input of this bidirectional LSTM is the word embeddings and the POS embeddings of the words in the sentence. Then, the output is passes to the next layer.

Entity detection is considered as a sequence labeling task in this paper, and so an entity tag is assigned to each word. A hidden layer and a softmax layer is used for entity detection and the input of this NN is the output of the sequential layer and the label embedding of the previous word.

The main focus of the dependency layer is the shortest path between the target words. A bidirectional tree structured LSTM is used to capture the dependency structure. In this bidirectional structure, the information is carried in both direction, which

is critical for relation extraction.

Three different structures are prepared: the shortest path structure (SP-Tree), SubTree, which is the subtree under the lowest common ancestor of the target words and FullTree, which is the full dependency tree.

To use the both sequence and dependency information, the input of the bidirectional tree structured LSTM consists of the output of the sequence layer, the dependency type embedding and the label embedding.

In the model, relations are represented with type and direction, and a pair has negative relation if the entities are wrong or the pair has no relation.

The relation candidate vector contains the hidden state vector of the top LSTM unit in the bottom up LSTM, and the hidden state vectors of the corresponding LSTMs of the target words. Then, these features are fed into neural network with one hidden layer and softmax layer. Since the impact of the sequential layer is indirect, the average of hidden state layers are concatenated to the features.

To reduce the effect of the bad prediction of the entities in the early stages, scheduled sampling and entity pretraining is applied. In scheduled sampling, the labels are corrected with a decaying probability.

The model was tested on ACE05 and ACE04 for end to end relation extraction and SemEval 2010 Task 8 for relation classification. In ACE05, there are 7 entity types and 6 relation types. In ACE04, there are 7 entity types and 7 relation types. In SemEval 2010 Task 8 dataset, there are 9 relation types and "Other" relation. The "Other" relation could be considered as a negative negative relation type.

The results of the end to end approach show that the proposed model performs better than the previous models. The results of the ablation tests on scheduled sampling and entitiy pretraining show that they both have positive effect of on the perfor-

mance of the system. The effect of the using shared parameters were also tested. The results show that separating the entity detection model and relation extraction model decreases the performance of the both models. The results of the SPTree, SubTree and FullTree are similar but if the shortest path is not distinguished, the performence of FullTree decreases.

The results from the relation classification show that the performance of the proposed network is comparable with the performance of the previous works. The results of SPTree, SubTree and FullTree are compared and FullTree performs worse than others. The results from the proposed network without the sequence layer show that the performance of the model decreases, and this means the sequence layer is necessary.

The next paper [5] is A Dependency-Based Neural Network for Relation Classification. The goal of this paper is to maximize the use of dependency information in sentences, and a new structure, called augmented dependency path (ADP) was proposed. To process the ADP, dependency-based neural networks (DepNN) are proposed, and it contains a recursive neural network to handle the subtrees and a convolutional layer to extract the most important features.

The effect of the dependency relations for relation classification has been already reported and there are different works about it. In the paper it is mentioned that "the most useful dependency information in relation classification includes the shortest dependency path and dependency subtrees". In this paper, a combination of these two components is proposed. The proposed structure is Augmented Dependency Path (ADP), which connects dependency subtrees to the corresponding words on the shortest dependency path. In the proposed neural network, one convolutional layer is used on the shortest dependency path, since CNNs are good at extracting the most useful features, and one recursive neural network is used on the dependency subtrees to extract the semantic representations, since RNN can model hierarchical structures. The words on the shortest path are combined with their subtrees to complete the neural network.

In the proposed network, each subtree of a word on the shortest path is processed by an RNN to compute a subtree embedding c_w, and c_w is concatenated to the word embeddings of the current word. These features are fed to a CNN to extract the combined features.

Dependency subtrees are used to get a good representation of the words on the shortest path, since the effect of each word can be derived from itself and its dependency subtree. For each word on subtree, the word embedding and the subtree embedding are concatenated. A dependency relation is added between each node of the subtree, and for each dependency relation, there is a corresponding dependency relation matrix, which will be learned during the training. Using the dependency relation, the embedding of the upper element in subtree can be computed so we can get the subtree embeddings of the words on the shortest path by applying this procedure to each word on the shortest path.

The subtree embeddings and the word embeddings of the words on the shortest words are concatenated and fed into a max pooling CNN to extract the sentence level features.

Some lexical level features such as named entity tags and WordNet hypernyms are also extracted and combined with the sentence level features. Then, these features are fed into a softmax layer to predict the relations between the nominal words.

In this paper, SemEval-2010 dataset was used. The dataset contains 8000 training and 2717 test sentences. There are 9 relation classes and "Other" relation class, which contains the sentences whose nominal words are not related with these 9 relation classes.

In the experiments, the effect of using dependency relations, attached subtrees and lexical features were tested. The results show that using each of the component increases the performance of the network and the final result can reach the state of the art results.

The next paper [6] is Fast and Large-scale Unsupervised Relation Extraction, which approaches the relation extraction problem in an unsupervised way.

A general way of relation extaction in an unsupervised way is clustering the patterns correspons to the same relation. There are the main challenges in this problem, the first one is finding the semantic representation of the relations and the last one is scalibility to large data.

This paper provides three main contributions. The first one is that a system for relation extraction, which is a practical and scalable to large data is constructed. The second one is the system uses approximations and the results are comparable to the one without approximations. The last one is a reasonable design for relations patters are suggested.

In the paper, first the entities are extracted in the dataset. A score value for each token is calculated and the token is decided as an entity if the score is above some threshold value. To calculate this score value, correlation and discount functions are used.

After entity extraction part, entity pairs are extracted. An entity pair is selected if the co-occurence of the entities is above some threshold value.

Then, relational patterns are extracted. Dependency relationship is used in this step. A triple is constructed as the dependency relations of the two entitesin the sentence and predicate of the sentence. A relational pattern is accepted if the frequency of the relation is above some threshold.

Pattern vectors are contructed. The values of the vector are the number of co-occurrences of the entities in the corresponding patterns. Two different statistical measures are used for co-occurrence, raw frequencies and PMI.

Then, since the vector space is high dimensional and sparse, a dimensionality

reduction technique is applied. In this paper, Principal Component Analysis (PCA) is used for dimensionality reduction.

In the paper, since it might be inefficient to count the exact number of co-occurrence of the entities for large data, an approximate counting method is proposed. The idea is it might be enough to find top-k entity pairs with larger counts of co-occurrences.

In the paper, the experiments are set to understand the effects of exact counting, approximate counting, PCA and word vectors. The results show that the results from the exact counting and the results from approximate counting are about the same.

The next paper [7] is Unsupervised Open Relation Extraction, which proposes a method which uses word embeddings to find the relational information in the text.

In this paper, a new method for relation extraction is proposed. In this method, word embeddings are re-weighted according to their position in the dependency path.

In the proposed method, there are four steps, preprocessing, feature extraction, sparse feature reduction and relation clustering.

In the preprocessing step, the named entities are extracted. Then, only the sentences which contain at least two named entities are selected. The lexicalized dependency path between the entity pairs are extracted using the Stanford CoreNlp dependency parser.

In the feature extraction, each sentence are transformed to vectors. These vectors are constructed using word embeddings, dependency paths between named entities and entity types. In the paper, a method which re-weights the word embeddings according to being inside the dependency paths between named entities and the length of the dependency paths between named entities is proposed.

In the sparse feature reduction, concatenating the the features from previous step for each relation might be a problem for clustering. Therefore, Principal Component Analysis (PCA) is used for dimensionality reduction.

In relation clustering, Hierarchical Agglomerative Clustering (HAC) is used to cluster the features.

NYT-FB dataset is used in the experiments. The experiments show that using this re-weighted embeddings and using only the dependency paths between named entities instead of whole sentence increases the results.

2.2. Related Datasets

One of the most common datasets for relation classification is SemEval 2010 Task 8. There are 8000 training instances and 2717 test intances. The relations in the dataset are Cause-Effect, Component-Whole, Entity-Destination, Entity-Origin, Product-Producer, Member-Collection, Message-Topic, Content-Container, Instrument-Agency and Other. The main evaluation metric used is macro-averaged F1 score.

One of the relation classification dataset is New York Times Corpus. This corpus is created for distantly supervised relationship extraction. The Stanford NER system is used to extract named entities.

TACRED is a relation extraction dataset with 106,264 instances. There are 41 relation types. The main evaluation metric used is micro-averaged F1 score.

Since there is no publicly available Turkish data for relation extraction, we used the Turkish data from wikipedia. We selected 25 politicians and 25 football players. Then, we collected whole text data of the politicians and the football players. In the data, there are 6626 sentences.

3. METHODS

We used Turkish text data from wikipedia. We selected 25 politicians and 25 football players and collected the whole text data from the his/her wikipedia page. The data contains 6626 sentences.

We applied two different unsupervised methods to extract the relation between the entities in a sentence. The first one was based on counting the number of co-occurrences of the entities. The second one used word embeddings and clustering algorithms.

3.1. Data Preparation

After we collected the data, we used ITU NLP tool to process the data.

In the preprocessing, we first splitted the sentences. After we got the sentences, we started to process each sentence. We got the dependency parse tree of each sentence. An example output of a dependency parse tree of sentence "Bütün insanlar hür , haysiyet ve haklar bakımından eşit doğarlar . ":

```
1       Bütün       bütün       Adj       Adj       _       2       MODIFIER
2       insanlar       insan       Noun       Noun       A3pl|Pnon|Nom ⌋
↪          10       SUBJECT
3       hür       hür       Adj       Adj       _       10       MODIFIER
4       ,       ,       Punc       Punc       _       3       PUNCTUATION
5       haysiyet       haysiyet       Noun       Noun       A3sg|Pnon|Nom ⌋
↪          7       COORDINATION
6       ve       ve       Conj       Conj       _       5       CONJUNCTION
7       haklar       hak       Noun       Noun       A3pl|Pnon|Nom       8 ⌋
↪          POSSESSOR
8       bakimindan
↪ bakim       Noun       Noun       A3sg|P3sg|Abl       10       MODIFIER
9       eşit       eşit       Adj       Adj       _       10       MODIFIER
```

```
10	dogarlar	dog	Verb	Verb	Pos|Aor|A3pl	⌟
↪ 0	PREDICATE
11	.	.	Punc	Punc	_	10	PUNCTUATION
```

Then, we extracted morphological analysis of the each word of each sentence. An example output of a morphological analysis of sentence "Bütün insanlar hür , haysiyet ve haklar bakımından eşit doğarlar . Akıl ve vicdana sahiptirler ve birbirlerine karşı kardeşlik zihniyeti ile hareket etmelidirler . ":

```
<S> <S>+BSTag
Bütün bütün+Adj bütün+Noun+NAdj+A3sg+Pnon+Nom Bütün+Noun+Prop+A3sg+Pnon+Nom
insanlar insan+Noun+A3pl+Pnon+Nom insan+Noun+A3sg+Pnon+Nom^DB+Verb+Zero+Pres+A3pl
hür hür+Adj hür+Noun+NAdj+A3sg+Pnon+Nom hür+Adverb
, ,+Punc
haysiyet haysiyet+Noun+A3sg+Pnon+Nom
ve ve+Conj
haklar hakla+Verb+Pos+Aor+A3sg hakla+Verb+Pos^DB+Adj+AorPart
↪ hak+Noun+A3pl+Pnon+Nom hak+Noun+A3sg+Pnon+Nom^DB+Verb+Zero+Pres+A3pl
bakımından bakım+Noun+A3sg+P2sg+Abl bakım+Noun+A3sg+P3sg+Abl bakımından+Adverb
eşit eşit+Adj eşit+Noun+NAdj+A3sg+Pnon+Nom
doğarlar doğ+Verb+Pos+Aor+A3pl
. .+Punc
Akıl ak+Verb+Pass+Pos+Imp+A2sg akıl+Noun+A3sg+Pnon+Nom
ve ve+Conj
vicdana vicdan+Noun+A3sg+Pnon+Dat
sahiptirler sahip+Noun+A3sg+Pnon+Nom^DB+Verb+Zero+Pres+A3pl+Cop
ve ve+Conj
birbirlerine birbiri+Pron+Quant+A3pl+P3pl+Dat
karşı karşı+Adj karşı+Noun+NAdj+A3sg+Pnon+Nom karşı+Adverb karşı+Postp+PCDat
kardeşlik kardeşlik+Noun+A3sg+Pnon+Nom kardeş+Noun+A3sg+Pnon+Nom^DB+Adj+Fitfor
zihniyeti zihniyet+Noun+A3sg+Pnon+Acc zihniyet+Noun+A3sg+P3sg+Nom
ile il+Noun+A3sg+Pnon+Dat ile+Conj ile+Postp+PCNom
hareket hareket+Noun+A3sg+Pnon+Nom
etmelidirler et+Verb+Pos+Neces+A3pl+Cop
. .+Punc
</S> </S>+ESTag
```

After this step, we extracted named entity tags using morphological analysis of the each word in the sentence. An example output of a named entity recognizer of a sentence "Başkan İbrahim Turhan , İzmir'de saat 14:00'te düzenlediği basın toplantısında İstanbul Menkul Kıymetler Borsası dün yakaladığı yüzde 17 yükseliş ve 1 milyar lira işlem hacmi ile kurulduğu 26 Aralık 1985 tarihinden bu yana en parlak günlerinden birisini yaşadı diye belirtti . ":

```
<DOC> <DOC>+BDTag                O
<S> <S>+BSTag                O
Başkan başkan+Noun+A3sg+Pnon+Nom                O
İbrahim İbrahim+Noun+Prop+A3sg+Pnon+Nom                B-PERSON
Turhan Turhan+Noun+Prop+A3sg+Pnon+Nom                I-PERSON
, ,+Punc                O
İzmir'de İzmir+Noun+Prop+A3sg+Pnon+Loc                B-LOCATION
saat saat+Noun+A3sg+Pnon+Nom                B-TIME
14:00'te 14:00'te+?                I-TIME
düzenlediği düzenle+Verb+Pos^DB+Adj+PastPart+P3sg                O
basın basın+Noun+A3sg+Pnon+Nom                O
toplantısında toplantı+Noun+A3sg+P3sg+Loc                O
İstanbul İstanbul+Noun+Prop+A3sg+Pnon+Nom                B-ORGANIZATION
Menkul menkul+Adj                I-ORGANIZATION
Kıymetler kıymet+Noun+A3pl+Pnon+Nom                I-ORGANIZATION
Borsası borsa+Noun+A3sg+P3sg+Nom                I-ORGANIZATION
dün dün+Adverb                O
yakaladığı yakala+Verb+Pos^DB+Adj+PastPart+P3sg                O
yüzde yüz+Noun+Num+A3sg+Pnon+Loc                B-PERCENT
17 17+Adj+Num                I-PERCENT
yükseliş yükseliş+Noun+A3sg+Pnon+Nom                O
ve ve+Conj                O
1 1+Adj+Num                O
milyar milyar+Adj+Num                O
lira lira+Noun+A3sg+Pnon+Nom                O
işlem işlem+Noun+A3sg+Pnon+Nom                O
hacmi hacim+Noun+A3sg+P3sg+Nom                O
ile ile+Conj                O
kurulduğu kurul+Verb+Pos^DB+Adj+PastPart+P3sg                O
26 26+Adj+Num                B-DATE
```

```
Aralık aralık+Noun+A3sg+Pnon+Nom                I-DATE
1985 1985+Adj+Num               I-DATE
tarihinden tarih+Noun+A3sg+P3sg+Abl                O
bu bu+Det               O
yana yan+Noun+NAdj+A3sg+Pnon+Dat                O
en en+Adverb                O
parlak parlak+Adj               O
günlerinden gün+Noun+A3pl+P3pl+Abl                O
birisini biri+Pron+Quant+A3sg+P3sg+Acc                O
yaşadı yaşa+Verb+Pos+Past+A3sg                O
diye diye+Postp+PCNom               O
belirtti belir+Verb+Caus+Pos+Past+A3sg                O
. .+Punc                O
</S> </S>+ESTag
<DOC> <DOC>+EDTag               O
```

After extracting named entities, we selected sentences which contains at least 2 entities and eliminated the other sentences.

3.2. Counting Methods

In the counting methods, we consider two different information about in the sentence, created triples for each of them and count their co-occurrences accordingly.

The first triple type was based on the entity types. The first element of the triple was the entity type of the first entity. The second element of the triple was the entity type of the second entity. The third element of the triple was the root of the predicate in the sentence which was extracted in the dependency parser. An example for this first triple type for the previous example was (PERSON, LOCATION, belir) .

The second triple type was based on the dependency relation of the entities in the sentence. The first element of the triple was the dependency relation of the first entity in the sentence. The second element of the triple was the dependency relation

of the second entity in the sentence. The third element of the triple was the root of the predicate in the sentence which was extracted in the dependency parser. An example for this first triple type for the previous example was (SUBJECT, MODIFIER, belir) .

Then, after counting these triples, we studied the distribution of the triples and set a threshold value to decide which triple definitions were valid for a relation type.

3.3. Clustering Methods

In the clustering methods, we first extracted some features. Features includes word embeddings of the each word in the sentence and entity types of the selected entities in the sentence.

We used a pretrained Turkish word embedding which is provided by fasttext [8]. This word embedding was trained on Common Crawl and Wikipedia and the dimension of each word vector was 300.

Since the dimentionality became very large for each sentence we used Principal Component Analysis (PCA) for dimentionality reduction.

After obtaining the features of each word in the sentence, we concatenated with the vectorized forms of the entity types of the selected entities in the sentence and formed the combined features.

Then, we used a clustering method to cluster the sentences using combined features. We decided to use Agglomerative Clustering and we used different cluster numbers while studying the results.

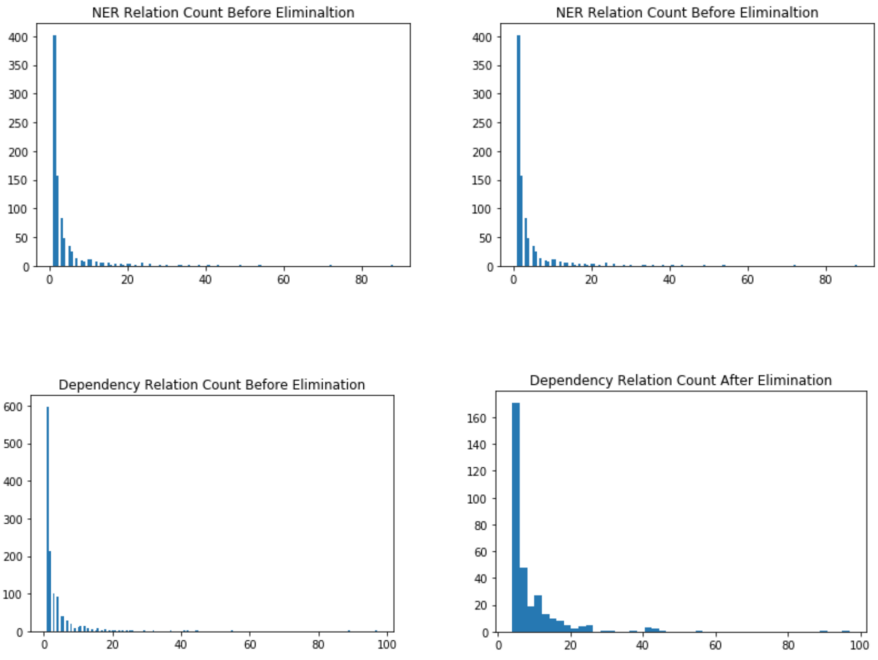
4. RESULTS

In this project, we selected 25 politicians and 25 football players. Then, we collected Turkish text data of the politicians and football players from wikipedia. There were 6626 sentences in total.

After we used ITU NLP tool api for the preprocessing part. The tool provides sentence splitter, dependency parser, morphological analyzer and named entity recognizer. We realized that the output of the tools did not look always correct, for instance, some of the named entities were not tagged, especially foreign person names. But still most of the outputs looked acceptable and we continued to the next step.

After we extracted named entities, we eliminated the sentences which contains less than two named entities and the number of the sentences which contains at least two named entities is 1481. So, we did not used three quarters of the data. Therefore, it might be better to use the text data of the more people from wikipedia.

Then we applied the counting methods. The results show that occurrences of the predicates affects the re-occurrences of the triples for each type of the triple. For instance the predicate "ol" was more often than the others and therefore, it was hard to understand the meaning of triple, since the triple set was too noisy. Also, we realized that some of the triples occur only a few times, so we decided to set a threshold according to the distribution of the occurrences and the averages of the named entity relation count and dependency relation count are 3.9 and 3.6. So we set 4 as a threshold and decided to accept a triple as a relation if the number of occurrence is greater than 4.

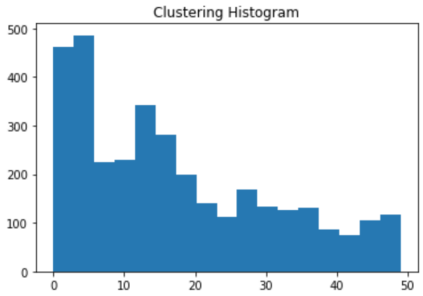


Some examples of the first triple type which was based on the entity types is ['DATE', 'LOCATION', 'kazan'] . The following sentences contains this triple. The first sentence is "2005 yılında Arjantin U-20 millî takımı ile Hollanda'da FIFA 20 Yaş Altı Dünya Kupası'nı kazandı." and the corresponding entities are "2005" and "Hollanda'da". The second example is "Final'de 9 Mayıs günü Bükreş'te Athletic Bilbao ile oynanan maçta'da 2 gol attı ve takımı maçı 3-0 kazandı." and the corresponding entities are "9 Mayıs" and "Bükreş'te". The third sentence is "Kaptanı olduğu Almanya millî futbol takımı ile 2014 Dünya Kupası'nı kazanmıştır." and the corresponding entities are "2014" and "Almanya". So, the first two triple looks like contains the information of "when" and "where", but in the last one "Almanya" was tagged in a wrong way as a location, the results looks acceptable. It might be also interesting to see the answer of "what" but the answer of this answer was not tagged as an entity. So, this might be a big downside of this method.

Some examples of the second triple type which was based on the the dependency relation of the entities in the sentence is ['MODIFIER', 'POSSESSOR', 'seç'] . The following sentences contains this triple. The first sentence is "2 Kasım 2004'te ABD Senatosuna seçildi." and the corresponding entities are "2 Kasım 2004" and "ABD". The second example is "4 Kasım 2008, 2008 ABD başkanlık seçimleri'nde ABD'nin

44. devlet başkanı seçildi." and the corresponding entities are "4 Kasım 2008" and "ABD". The third sentence is "2012 Avrupa Futbol Şampiyonası'nda ise millî formayla 2. kez Avrupa şampiyonluğu yaşadı ve turnuvanın en iyi oyuncusu seçildi." and the corresponding entities are "2012" and "Avrupa". So, the first two triple looks like contains a similar relationship, but it is hard to accept that the entities in the last have a similar relation as the others.

Also, we conducted some experiments with clustering methods. We used pretrained word embeddings from fasttext [8] and the dimension of a word vector is 300. Then, we created sentence vectors using pretrained word embeddings. Since each sentence might contain different number of words, we took the average of the word embeddings and got a fixed sized feature for each sentence. Since the dimension of the word embeddings is high, we used Principal Component Analysis (PCA) to reduce the dimensions of the features. Then, we created entity type embeddings and initialized randomly. This random initialization provides a better perspective than the one hot encodings. After that, we concatenated sentence features and entity type embeddings and got combined features. We clustered the sentences with using Agglomerative Clustering method. The histogram of the clusters is as the following.



The results show that taking only the averages and entity types causes duplicate results if the sentence has more than two entities and they are the same type. So additional features should be added to diverge them.

Also, the evaluation of this method was not trivial. The clusters have different types of entity pairs and it was hard to understand and interpret the similarity between

the relations between entity pairs.

5. CONCLUSION AND DISCUSSION

There are lots of work on relation extraction especially in English. First attempts are done by hand crafted features which require domain specific knowledge and these methods were hard to generalize the general relation extraction problem and their performance could vary on different domains. Neural networks are a way of extracting features and these models are not domain specific and can be generalized more easily. So recent works about relation extraction are mainly focused on these type of methods. The main structure of these methods could be considered as extracting some sentence level features, extracting some lexical features and feeding them into some classifier to decide what the relation between nominal words is. Although the main structure is similar, there are different ideas for each step. For instance, one of the methods uses CNNs to extract the sentence level features and one uses LSTMs. Also, some works mentioned that using some NLP methods, such as NER and dependency parse trees, could be useful and contains relevant information about the relation between the nominal words. Therefore, we will apply these different methods and compare the affects of the each component.

In this project, we decided to study the relation extraction on the annotated data and used both of the counting methods and clustering methods which use word embeddings. The results show that there are pros and cons of these methods. Although we used a relatively small data for the unsupervised methods, we could find some meaningful relations. Therefore, this methods looks promising and the results might become better with some additions and better tools.

6. FUTURE WORK

In this project, we used some unsupervised approaches to find the relation between entities. The results show that the data should be bigger and the tools used for named entity recognizer have some critical role in this project. So, the upcoming works should use on the unsupervised relation extraction problem with bigger data. Also, it might be interesting to apply the supervised relation extraction methods mentioned in the Literature Section, such as CNN's and LSTM's if enough annotated Turkish data is provided.

REFERENCES

1. Zeng, D., K. Liu, S. Lai, G. Zhou and J. Zhao, "Relation Classification via Convolutional Deep Neural Network", *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 2335–2344, Dublin City University and Association for Computational Linguistics, Dublin, Ireland, Aug. 2014, https://www.aclweb.org/anthology/C14-1220.

2. dos Santos, C., B. Xiang and B. Zhou, "Classifying Relations by Ranking with Convolutional Neural Networks", *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 626–634, Association for Computational Linguistics, Beijing, China, Jul. 2015, https://www.aclweb.org/anthology/P15-1061.

3. Zhang, S., D. Zheng, X. Hu and M. Yang, "Bidirectional Long Short-Term Memory Networks for Relation Classification", *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pp. 73–78, Shanghai, China, Oct. 2015, https://www.aclweb.org/anthology/Y15-1009.

4. Miwa, M. and M. Bansal, "End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures", *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1105–1116, Association for Computational Linguistics, Berlin, Germany, Aug. 2016, https://www.aclweb.org/anthology/P16-1105.

5. Liu, Y., F. Wei, S. Li, H. Ji, M. Zhou and H. Wang, "A Dependency-Based Neural Network for Relation Classification", *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 285–290, Association for Computational Linguistics, Beijing, China, Jul. 2015,

https://www.aclweb.org/anthology/P15-2047.

6. Takase, S., N. Okazaki and K. Inui, "Fast and Large-scale Unsupervised Relation Extraction", *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pp. 96–105, Shanghai, China, Oct. 2015, https://www.aclweb.org/anthology/Y15-1012.

7. Elsahar, H., E. Demidova, S. Gottschalk, C. Gravier and F. Laforest, "Unsupervised Open Relation Extraction", E. Blomqvist, K. Hose, H. Paulheim, A. Ławrynowicz, F. Ciravegna and O. Hartig (Editors), *The Semantic Web: ESWC 2017 Satellite Events*, pp. 12–16, Springer International Publishing, Cham, 2017.

8. Grave, E., P. Bojanowski, P. Gupta, A. Joulin and T. Mikolov, "Learning Word Vectors for 157 Languages", *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

APPENDIX A: DATA AVAILABILITY STATEMENT

- The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.