Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 2388 – 2398

doi:10.3906/elk-1512-102

Turkish entity discovery with word embeddings

Murat KALENDER[1,2,*], **Emin Erkan KORKMAZ**[1]

[1]Department of Computer Engineering, Faculty of Engineering, Yeditepe University, İstanbul, Turkey

[2]Department of Technology Introduction, Huawei Technologies, İstanbul, Turkey

Received: 11.12.2015 • **Accepted/Published Online:** 20.09.2016 • **Final Version:** 29.05.2017

Abstract: Entity-linking systems link noun phrase mentions in a text to their corresponding knowledge base entities in order to enrich a text with metadata. Wikipedia is a popular and comprehensive knowledge base that is widely used in entity-linking systems. However, long-tail entities are not popular enough to have their own Wikipedia articles. Therefore, a knowledge base created by using Wikipedia entities would be limited to only popular entities. In order to overcome the knowledge base coverage limitation of Wikipedia-based entity-linking systems, this paper presents an entity-discovery system that can detect semantic types of entities that are not defined in Wikipedia. The effectiveness of the proposed system was validated empirically through the use of generated data sets for the Turkish language. The experimental results show that, in terms of accuracy, our system performs competitively in comparison to the previous methods in the literature. Its high performance is achieved through a method that learns word embeddings for candidate entities.

Key words: Entity discovery, fine-grained entity types, entity linking, word vectors, deep neural networks, knowledge base population

1. Introduction

The amount of unstructured data has increased exponentially in recent years and Web resources form the vast part of it, including Tweets, blogs, online news, and comments. Leveraging these resources through automatic processing techniques is highly challenging due to the ambiguity of natural language [1]. The data need to be transformed into a standard format that contains metadata so that they can be used in different information retrieval and extraction applications, such as semantic search, question answering, and summarization systems.

Entity linking is one of the problems to be handled in order to process natural language and to enrich the existing unstructured text with metadata. The generation of assignments between knowledge base entities and lexical units is called entity linking. For example, Figure 1 shows an example mapping of a piece of textual content to entities defined in Turkish Wikipedia (Vikipedi, tr.wikipedia.org) as of 26 August 2015. A spotter would detect the two entity mentions defined in Wikipedia ("Arsenal" and "passing") in this sentence.

The success of an entity-linking system clearly depends on the term coverage of the knowledge base utilized. In order to effectively process domain-independent documents, a comprehensive, up-to-date, and evolving knowledge base is required. Wikipedia is a popular knowledge base that satisfies these requirements and is therefore widely used in entity-linking systems. However, long-tail entities are not popular enough to have their own Wikipedia articles. For example, an article for "Yetenek Sizsiniz Türkiye" (the Turkish version

*Correspondence: murat.kalender@huawei.com

Figure 1. A linking of a piece of textual content to entities defined in a knowledge base.

of the "Got Talent" television show series) exists in Vikipedi and it is typed as a television show; however, "İbo Show", another famous Turkish television show, lacks a Vikipedi article (date of access: 26 August 2015). In this study, an entity-discovery system is proposed that semiautomatically detects entity mentions without a Vikipedi entry in a given Turkish text corpus. The system also semantically types detected unlinkable entity mentions. For informative knowledge, we aim to type new entities in a fine-grained manner (e.g., "basketball player", "economist", "airport", as opposed to generic types like "person", "organization", "event") [2].

There are two main aspects of an entity-discovery system: the detection of candidate entities and the prediction of their semantic types. We address the first part by using an n-gram-based approach to detect frequent noun phrases in a given corpus as candidate entities. The second part is addressed with a fine-grained entity recognizer. There are three main challenges in creating a fine-grained entity recognizer: selection of the types, creation of the training data, and development of a fast and accurate multiclass classification algorithm [3]. We address the first challenge by collecting a set of 100 unique tags (types), which were extracted from Vikipedi articles' infobox (https://en.wikipedia.org/wiki/Help:Infobox) information. The second challenge, creating a training set for these tags, is addressed by utilizing existing Vikipedi article content and word vectors (embeddings). Vikipedi articles with a description and tag information are used for the training process. Each article description is parsed sentence by sentence and a set of words (verbs and nouns) is extracted from the text. The extracted set of words is then converted into a set of vectors using the corresponding word vectors. Afterwards, the set of vectors is converted into a fixed-length feature vector through average pooling [4]. Finally, the labeled training data are used to train a linear classifier model for the fine-grained entity recognizer.

The generation of entity vectors based on word vectors and their classification using linear classifiers form the core part of the proposed approach. The effectiveness of the system is validated empirically by using evaluations over generated data sets. The experimental results indicate that our system performs competitively compared to previous methods in terms of accuracy. The main contributions of this paper are summarized as follows.

- A novel Turkish entity-discovery system is proposed that semantically types entities not defined in Wikipedia.
- The effectiveness of various feature combinations were tested for the fine-grained entity recognition task.
- A new data set was created for evaluating the performance of the proposed Turkish entity-discovery system. The experimental results show that our system can achieve high accuracy on the data set.

2. Related work

This section first provides background information about the word embeddings utilized in this study. Existing studies on fine-grained entity recognition and Turkish natural language processing (NLP) systems are then discussed.

2.1. Word embedding

Word embedding is a distributed representation of a word that utilizes a high dimensional vector, where each dimension corresponds to a latent feature of the word [5]. Thus, word embedding can capture both the semantic and syntactic information of the corresponding word. The resulting distributed representation has several advantages compared to traditional language models, such as bag-of-words (BOW), in terms of compactness and sparsity. Also, semantically similar words are represented with closer vectors. For example, two semantically similar words, such as "capital" and "city", are expected to have similar values at least in some dimensions in their corresponding vectors.

Word2Vec [6] and GloVe [7] are two popular word-embedding algorithms used to construct vector representations for words. Word2Vec (http://code.google.com/p/word2vec/) and GloVe (http://nlp.stanford.edu/projects/glove/) are both open source and publicly available tools. Also, pretrained word vectors for English words are provided in these websites.

2.2. Fine-grained entity recognition

Fine-grained entity recognition is the task of identifying semantic types of entities in the text. A key feature of named entity recognition (NER) is that more specific entity types are used in the fine-grained entity recognition process. For example, "basketball player" is an entity type that would be used for typing basketball players, such as Michael Jordan, and is a more informative type than "person".

In contrast to coarse-grained NER [8], there are fewer fine-grained entity recognition studies [2,3,9–12] proposed in the literature. These studies utilized trained classifiers over a variety of linguistic features (i.e. part-of-speech tags, unigrams, bigrams) and contextual features (preceding and following words). Specifically, Ling et al. [3] proposed FIGER, which classifies entity mentions with 112 unique tags curated from Freebase [13] types. They trained a conditional random field by utilizing Wikipedia anchor links as training data. Yosef et al. [11] proposed HYENA, which is a multilabel classifier based on the hierarchical taxonomy of the YAGO [14] knowledge base. In this study, a support vector machine (SVM)-based classifier was utilized on Wikipedia anchor links, similar to the work of Ling et al. [3].

The word embeddings approach is also applied to the fine-grained entity recognition task. Recently, Yogatama et al. [15] proposed a novel method to learn an embedding for each entity type and each feature. This way, feature vectors could be created for entity mentions in order to classify entities. They compared their entity recognition method with FIGER and observed better performance (72.35% F1 score).

In contrast to many successful applications for English, there is currently no publicly available fine-grained entity recognition system for Turkish, though there are some studies [16,17] proposed to solve the coarse-grained NER task.

2.3. Turkish NLP systems

There are currently two publicly available popular NLP tools for Turkish, Zemberek [18] and the ITU Turkish NLP Web Service [19].

Zemberek is a popular open source NLP library for Turkish. Zemberek provides the most commonly used NLP tasks, such as sentence detection, tokenization, morphological analysis, and morphological disambiguation.

The ITU Turkish NLP Web Service is another publicly available NLP library, which operates on a "software as a service" basis and provides state-of-the-art NLP tools in many layers: preprocessing, morphology, syntax, and entity recognition.

In this study, sentence detection, morphological parsing, and disambiguation processes were carried out with the Zemberek library functions instead of the ITU Turkish NLP Web Service, mainly because of runtime performance considerations.

3. Turkish entity discovery

This section introduces our entity-discovery system for Turkish language. The system semiautomatically detects entity mentions that are not defined in Vikipedi for a given Turkish text and semantically types the detected unlinkable entity mentions.

As shown in Figure 2, the Turkish entity-discovery system was developed through the design and implementation of three major modules: a candidate entity detector, a feature extractor, and a fine-grained entity recognizer.

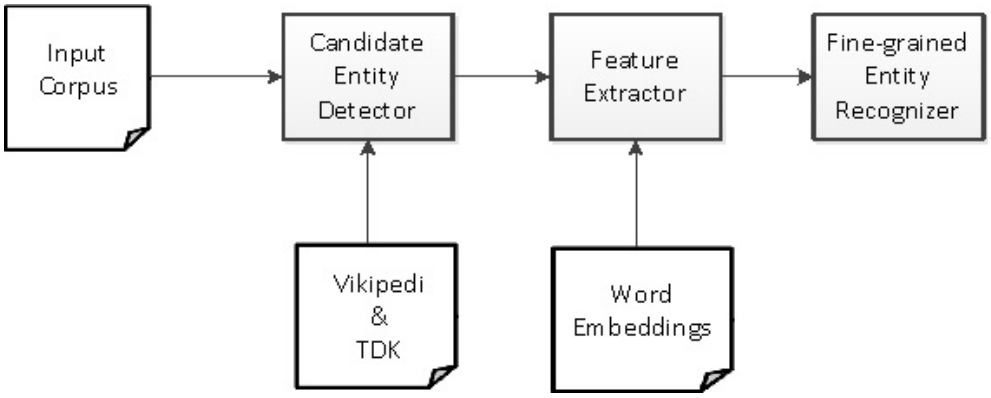

Figure 2. Turkish entity-discovery system architecture.

3.1. Candidate entity detector

The candidate entity detector module produces a list of possible entity mentions for a given corpus. Entity mention refers to small fragments of text that may correspond to an entity in a given knowledge base. This task is achieved in two steps.

In the first step, the candidate entity detector produces all possible n-grams (where n can be between 1 and 4) of successive nouns in a sentence. This task consists of 1) sentence detection, where each document in the

corpus is split into sentences; 2) lemmatization and parts of speech detection, where each word is analyzed to find its root and part-of-speech tag; and 3) finding noun phrases, where groups of successive nouns are identified. The idea behind this approach is that multiword entities, especially special names, location names, etc., usually consist of noun phrases.

In the second step, the candidate entity detector identifies frequently occurring noun phrases as candidate entities and filters the ones that are already defined in the utilized knowledge base and dictionary. In this study, all phrases that occur in either Vikipedi or TDK(the official Turkish dictionary; http://www.tdk.gov.tr/) were filtered out, since they are already known entities for Turkish.

3.2. Feature extractor

The feature extractor module takes the input list of entity mentions and the list of sentences in which the entities occur from the candidate entity detector module. Then an entity vector is produced for each entity mention by this module. This task is achieved in two steps.

In the first step, the feature extractor extracts linguistic features (part-of-speech tags and suffixes) and contextual features through morphological analysis and the morphological disambiguation functionalities of the Zemberek NLP tool. Morphological analysis is done for each sentence: the suffixes of the candidate entity mention are identified and part-of-speech tags of other words in the sentence are determined. Morphological analysis of a word may result in more than one possible parsing due to ambiguity. For example, there are four possible analyses for the Turkish word "kalemi", as shown in Figure 3. To handle morphological ambiguity, morphological disambiguation is applied after the morphological analysis, again by using the Zemberek NLP tool.

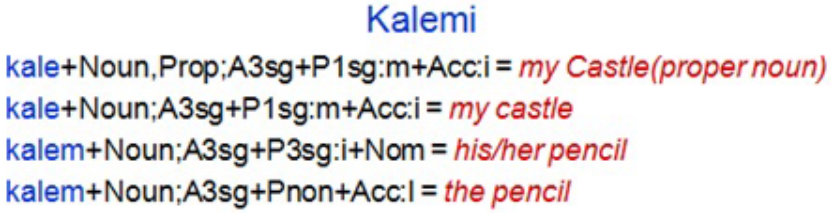

Figure 3. Zemberek NLP library morphological analysis.

Extracted part-of-speech tags are used to categorize and filter contextual information of the entity mentions. Contextual words besides nouns and verbs (e.g., adverbs, adjectives, prepositions) are filtered since they are not discriminating features. As a result of this step, a set of words and suffixes is created for each candidate entity. The table in Figure 4 shows the generated features for the given Vikipedi article "Yeditepe University". Note that Vikipedi articles' descriptions are used for creating feature vectors for known entities in order to obtain training data for the classification algorithm. The details of this process are provided in Section 3.3.

In the second step, extracted feature sets (title, nouns, verbs, and suffixes) for the entity mentions are used for creating vectors that represent the entities. In order to achieve this goal, Word2Vec word embedding and average pooling algorithms were used.

The Word2Vec word embedding algorithm was used to construct vector representations for words. Vikipedi articles content and the Milliyet corpus [20] were used as input data to cover all entities (nouns and verbs) as much as possible. The input data were also lemmatized to find the root form of Turkish words before training. Essentially, lemmatization makes the input space denser and disregards different vectors for

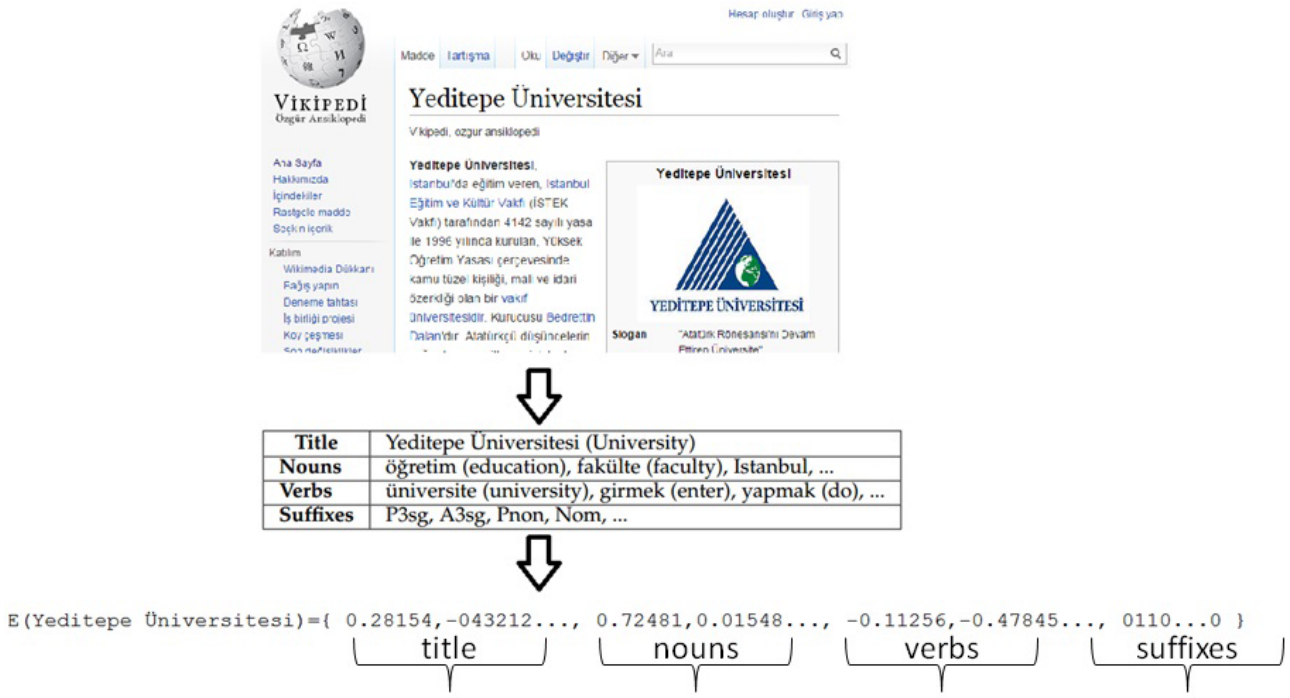

Figure 4. Feature vector extraction for the entity "Yeditepe University".

inflectional forms of words. After constructing word vectors, the extracted feature set (title, nouns, and verbs) for a candidate entity is converted into a set of vectors using the corresponding word vectors obtained by the Word2Vec algorithm.

Certainly, the number of features extracted for a candidate entity would differ based on the number of verbs and nouns that exist in the context where this entity appears. Average pooling algorithms [4] are applied in order to convert the set of word vectors into a fixed-length vector that can be processed in the fine-grained entity recognizer module to classify entities by their types. The algorithm simply takes the average of the word vectors and computes a fixed-length vector. The average pooling of set of word vectors can be computed as in the following formula:

$$\bar{x} = \frac{1}{N} \times \sum_{i}^{N} x_i,$$

where x_i denotes the word vector of the nth element in a given feature set and N is the number of word vectors.

In contrast to other features, suffixes are not full words. In order to represent suffixes in a vector format, the BOW method is utilized. As a result of this method, suffixes can be represented with a 64-length vector. To derive the final entity vector, all feature vectors are unified, as shown in Figure 4.

4. Fine-grained entity recognizer

A fine-grained entity recognizer is used for detecting semantic typing of entities. This module is realized in three steps: selection of the semantic types that will be utilized, creation of training data, and development of a fast and accurate multiclass classification algorithm [3].

ada	dağ	havalimanı	nehir	sporcu
albüm	deprem	havayolları	ödül	stadyum
amfibi	dergi	hükümdar	okul	sunucu
antlaşma	dil	hükümet kurumu	opera	sürüngen
asker	dil ailesi	işletim sistemi	örümcek	takımyıldız
askeri birim	din	karakter	otomobil	televizyon
ateşli silah	dini yapı	kimya	öykü	tenis sporcu
balık	dizi	kişi	oyun	tv kanalı
baraj	edebiyat	kitap	oyuncu	uçak
basketbol	film	köpek ırkı	programlama dili	ülke
basketbol kulübü	filozof	kuruluş	radyo istasyonu	üniversite
basketbol ligi	futbol kulübü	kuş	roman	video oyunu
basketbolcu	futbolcu	makam sahibi	sanatçı	voleybol kulübü
bayrak	galaksi	manken	şarkı	voleybolcu
bilim adamı	gazete	marş	savaş	website
bitki	gemi	memeli	seçim	yapı
biyoloji	göl	mitoloji	şirket	yazar
böcek	grup	müze	siyasetçi	yazılım
buz patencisi	güreşçi	müzik	siyasi makam	yerleşim
cep telefonu	hastalık	müzik sanatçısı	siyasi parti	yol

Figure 5. List of 100 curated tags used in this study.

The first step in entity recognition is defining the set of semantic types. Although there have been several studies [21] to create a standard comprehensive tag set, no consensus has been reached by the research community [9]. On the other hand, a collaborative knowledge base, such as Wikipedia, provides hundreds of types that are used to annotate each article in the website. Compared to other type sets and knowledge bases, the most important advantage of Wikipedia is that it contains a variety of entity types and that a high number of defined entities annotated with these types exist in this knowledge base. This is also true for the Turkish language. While Wikipedia tags are comprehensive, there are also some very specific types that need to be filtered for the process (e.g., Turkish village, football league season). To achieve a high-quality tag set, entity types were sorted by the number of entities annotated with them. The most frequently occurring types were manually analyzed by human experts and unnecessary ones were filtered. In the end, the 100 most frequent entity types were used as the tag set in this study (listed in Figure 5).

The second step, creating a training set for these tags, is achieved by annotating the content of Vikipedi articles (https://dumps.wikimedia.org/trwiki/20150826/) with a type in our tag set. Eligible articles' contents are given to the feature extractor module and a vector is created for each article. Here, the feature extraction procedure described in the previous section is used to form the feature vectors of Vikipedi entities. For the candidate entities, the procedure extracts the features from the context where the entity appears. For the Vikipedi entities, the content of the corresponding article is utilized, this time for the feature extraction process. As a result of this operation, 36,245 vectors are created. Since the Vikipedi articles have defined types, the data set formed can be utilized as a labeled set. In the final step, the labeled data are used to train classifier models for the fine-grained entity recognizer. Let $x = (x_1, x_2, x_3, ..., x_n)$ be the feature vector of the sequence of m instances, and let $y = (y_1, y_2, y_3, ..., y_n)$ be the vector of labels of the m instances. This multiclass classification problem can be solved by using classic linear classifiers in the form of:

$$\hat{y} = \frac{\text{argmax } \omega^T \times f(x, y)}{y}$$

where $\hat{y}$ is a predicted label, $f(x, y)$ is the feature vector of a mention x with a label $y \in T$, $T$is our set of 100 unique tags (types), and w is the weight vector of the model.

SVM, logistic regression, and Softmax classifiers were built as linear classifiers. SVM and logistic regression classifiers were created using Liblinear [22], which is a library for large linear classification. Softmax classifier was created using the software Encog [23]. Two parameters were set in Liblinear: cost of constraints violation = 2.0 (default: 1.0) and stopping criterion = 0.05 (default: 0.1). The cost of constraints violation was doubled and a smaller stopping criterion was used in order to learn a more accurate model over the training data utilized. We formed a simple three-layer neural network, which consisted of an input layer, a hidden layer, and an output layer. The number of neurons of the hidden layer was set to half of the input size and the number of neurons of the output (Softmax) layer was set as the number of tags, which is 100.

5. Evaluations and experiments

To evaluate the performance of the proposed entity discovery algorithm, the Vikipedi articles and a corpus formed from the Milliyet online newspaper were used. Vikipedi articles were used as a validation set in order to tune the parameters of the fine-grained entity recognizer. As described in Section 3.3, a set of labeled data with 36,245 instances was created by using Vikipedi articles. The data set was split into two parts: 70% training, 30% testing. The performances of the entity discovery algorithms were computed as in the following formula:

$$\text{accuracy} = \frac{\#\ \text{correct predictions}}{\#\ \text{total entities}}$$

First, the fine-grained entity recognizer was evaluated with different classifier algorithms and with varying word vector sizes on the Vikipedi data set. In the experiments, the linear classifiers SVM, logistic regression, and Softmax were utilized and the vector sizes (from 50 to 200) were tested. The performances of the linear classifiers were evaluated based on L2-regularized L2-loss support vector classification and L2-regularized logistic regression, which were implemented through a Java version of Liblinear (https://github.com/bwaldvogel/liblinear-java). Table 1 shows the performance values for this experiment. The SVM and larger vector sizes resulted in better performance values. Hence, the best performance result (78.69%) was obtained with the SVM and a vector size of 200. This setting was used for further experiments to assess the final performance of the system.

Table 1. Evaluation of the fine-grained entity recognizer algorithm with varying classifiers and word vector sizes.

Language	Vector size	SVM (%)	Logistic regression (%)	Softmax (%)
Turkish	50	74.16	73.15	73.30
	100	77.08	75.90	74.28
	200	78.69	77.51	74.75
English	200	77.14	-	-

We also evaluated our fine-grained entity recognition algorithm for the English language in order to prove that the approach is language-independent. In a similar way, English Wikipedia articles were processed and, again, the 100 most frequently occurring entity types were determined and the experimental data set was formed. By using an English NLP tool (https://opennlp.apache.org/) and the Glove word vectors for English, entity vectors were also created for English Wikipedia articles. We observed 77.14% accuracy for English, which is a very close performance compared to the experimental results in Turkish.

After tuning the parameters, the performance of each feature set (title, nouns, verbs, and suffixes) was calculated separately to assess their contributions. Table 2 shows the performance values for this experiment. The best performance was observed when all of the features were utilized together.

Table 2. Comparison of the performance values of the fine-grained entity recognizer with varying feature sets.

Features	Suffixes	Verbs	Title	Nouns	Verbs Title Nouns	All
Accuracy (%)	11.58	48.23	58.98	65.68	78.18	78.69

In the last experiment, the corpus [20] created by the Bilkent Information Retrieval Group (http://www.cs.bilkent.edu.tr/∼canf/bilir_web/) was used. The corpus contains 408,305 documents; they are news articles and columns collected from the Turkish newspaper Milliyet (www.milliyet.com.tr) from 2001 to 2005.

First, the Milliyet corpus was given as input to the candidate entity detector module and a list of candidate entities was extracted. This process produces hundreds of 1-grams, 2-grams, 3-grams, and 4-grams as candidate entities. However, these identified candidate entity mentions are noisy; not all of them are real entities. Concurrent entity mentions, typos, and HTML tables are the main causes of this problem. Therefore, human intervention is needed to finalize the candidate entity detection process. In the end, we manually selected and annotated 150 candidate entities as the test data. Samples from the manually annotated entities are listed in Table 3; column 1 contains entity titles and column 3 contains the manually assigned type information.

Table 3. Fine-grained entity recognition sample results of the Milliyet test data.

Candidate entity	Prediction	Correct
İbo Show	Televizyon (TV)	Televizyon (TV)
Polat Renaissance Otel	Yapı (construction)	Yapı (construction)
Van Hooijdonk	Futbolcu (footballer)	Futbolcu (footballer)
Pablo Montoya	Otomobil (automobile)	Sürücü (driver)
Sunday Times	Gazete (newspaper)	Gazete (newspaper)
Robert Pearson	Makam sahibi (officeholder)	Makam sahibi (officeholder)
Albert Einstein	Bilim adamı (scientist)	Bilim adamı (scientist)
Harun Doğan	Person	Sporcu (sportsman)

Finally, the manually created Milliyet data set was evaluated by the fine-grained named entity recognizer. The experiment resulted in 56.00% accuracy for strict typing of entities and 72.00% for relaxed typing of entities. In contrast to strict typing, classifying entities with a more general type is evaluated as a correct assignment in relaxed typing, such as "sportsman" instead of "footballer".

Note that the accuracy of this second experiment is lower compared to the first one. In fact, this is an expected result, since the classifier was trained using Vikipedi pages and then tested with different entities, but again collected from Vikipedi in the first experiment. However, in the second experiment, a general corpus collected from an online newspaper was utilized. Moreover, missing types and types from the same domain reduce performance. For example, "Pablo Montoya" was classified as an automobile rather than as a driver in the experiments. The reasons for this incorrect assignment are that "driver" was not in our tag set and that

driver and automobile entities are related entities that share similar contextual features (words). Hence, such a misclassification occurs in the system. In order to handle this kind of incorrect assignment, more sophisticated features and algorithms are needed. For example, a hierarchical classifier based on the taxonomy of Wikipedia types could be used to differentiate upper-level classes with higher accuracy. It would be easier to differentiate "person" and "machine (device)" rather than "driver" and "automobile".

6. Discussion and conclusion

This study presents an entity discovery system that semantically types entities not defined in Wikipedia. Fine-grained entity recognition is the key challenge in the proposed system. To address this problem, a supervised multiclass classification algorithm that leverages word-embedding models is proposed. Moreover, a series of experiments is conducted to evaluate the performance of the system. Evaluations show that the proposed system has a satisfactory accuracy, with 56.00% for strict typing of entities and 72.00% for relaxed typing of entities. Although there are slightly better performance results (72.35% F1 score) for English, there is no such study for the Turkish language to perform comparative evaluations.

Our research differs from the previous entity discovery studies [2,3,9–12] introduced in Section 2.2 mainly in three points. First, we used a full text corpus rather than a single document or plain text to discover entities and their types. Second, we applied the average pooling method to neural network-based language models to obtain feature vectors for candidate entities instead of using handcrafted features. Finally, our approach is simple to implement for other languages. As of 26 August 2015, there were 282 active Wikipedias (https://en.wikipedia.org/wiki/List_of_Wikipedias) in various languages that could be used for entity discovery.

In conclusion, this paper has demonstrated the potential of word vectors for entity discovery. In the future, the proposed entity discovery system could be fully automated by employing machine learning algorithms rather than using only the n-gram approach for candidate entity detection. Moreover, fine-grained entity recognizer performance could be improved by utilizing more sophisticated features and algorithms, such as a hierarchical classifier based on the taxonomy of Wikipedia types.

References

[1] Shen W, Wang J, Han J. Entity linking with a knowledge base: issues, techniques, and solutions. IEEE T Knowl Data En 2015; 27: 443-460.

[2] Nakashole N, Tylenda T, Weikum G. Fine-grained semantic typing of emerging entities. In: ACL 2013 51st Annual Meeting of the Association for Computational Linguistics; 4–9 August 2013; Sofia, Bulgaria. pp. 1488-1497.

[3] Ling X, Weld DS. Fine-grained entity recognition. In: 26th AAAI Conference on Artificial Intelligence; 22–26 July 2012; Toronto, Canada. Palo Alto, CA, USA: AAAI Press. pp. 94-100.

[4] Xing C, Wang D, Zhang X, Liu C. Document classification with distributions of word vectors. In: APSIPA 2014 Asia-Pacific Signal and Information Processing Association Conference; 9–12 December 2014; Siem Reap, Cambodia. New York, NY, USA: IEEE. pp. 1-5.

[5] Luong T, Socher R, Manning CD. Better word representations with recursive neural networks for morphology. In: CoNLL 2013 Computational Natural Language Learning Conference; Sofia, Bulgaria. pp. 104-113.

[6] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. In: ICLR 2013 International Conference on Learning Representations; 2–4 May 2013; Scottsdale, AZ, USA.

[7] Pennington J, Socher R, Manning CD. Glove: global vectors for word representation. In: EMNLP 2014 Empirical Methods in Natural Language Processing Conference; 25–29 October; Doha, Qatar. pp. 1532-1543.

[8] Nadeau D, Sekine S. A survey of named entity recognition and classification. Linguisticae Investigationes 2007; 30: 3-26.

[9] Lin T, Mausam, Etzioni O. No noun phrase left behind: detecting and typing unlinkable entities. In: EMNLP-CoNLL 2012 Empirical Methods in Natural Language Processing and Computational Natural Language Learning Conference; 12–14 July 2012; Stroudsburg, PA, USA. pp. 893-903.

[10] Rahman A, Ng V. Inducing fine-grained semantic classes via hierarchical and collective classification. In: COLING 2010 23rd International Conference on Computational Linguistics; 23–27 August 2010; Stroudsburg, PA, USA. pp. 931-939.

[11] Yosef MA, Bauer S, Hoffart J, Spaniol M, Weikum G. Hyena: hierarchical type classification for entity names. In: COLING 2012 24th International Conference on Computational Linguistics; 8–15 December 2012; Mumbai, India. p. 1361.

[12] Desmet B, Hoste V. Fine-grained Dutch named entity recognition. Lang Resour Eval 2014; 48: 307-343.

[13] Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J. Freebase: a collaboratively created graph database for structuring human knowledge. In: ACM SIGMOD 2008 International Conference on Management of Data; 9–12 June 2008; Vancouver, Canada. New York, NY, USA: ACM. pp. 1247-1250.

[14] Hoffart J, Suchanek FM, Berberich K, Weikum G. Yago2: a spatially and temporally enhanced knowledge base from Wikipedia. Artif Intell 2013; 194: 28-61.

[15] Yogatama D, Gillick D, Lazic N. Embedding methods for fine grained entity type classification. In: ACL-IJCNLP 2015 53rd Annual Meeting of the Association for Computational Linguistics and 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing; 26–31 July 2015; Beijing, China. pp. 291-296.

[16] Seker GA, Eryigit G. Initial explorations on using CRFs for Turkish named entity recognition. In: COLING 2012 24th International Conference on Computational Linguistics; 8–15 December 2012; Mumbai, India. pp. 2459-2474.

[17] Tatar S, Cicekli I. Automatic rule learning exploiting morphological features for named entity recognition in Turkish. J Inf Sci 2011; 37: 137-151.

[18] Akin AA, Akin MD. Zemberek, an open source NLP framework for Turkic languages. Structure 2007; 10: 1-5.

[19] Eryigit G. ITU Turkish NLP Web Service. In: EACL 2014 14th Conference of the European Chapter of the Association for Computational Linguistics; 26–30 April 2014; Gothenburg, Sweden. pp. 1-4.

[20] Can F, Kocberber S, Balcik E, Kaynak C, Ocalan HC, Vursavas OM. Information retrieval on Turkish texts. J Am Soc Inf Sci Technol 2008; 59: 407-421.

[21] Sekine S. Extended named entity ontology with attribute information. In: LREC 2008 6th International Conference on Language Resources and Evaluation; 28–30 May 2008; Marrakech, Morocco. pp. 52-57.

[22] Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ. Liblinear: a library for large linear classification. J Mach Learn 2008; 9: 1871-1874.

[23] Heaton J. Encog: library of interchangeable machine learning models for Java and C#. J Mach Learn Res 2015; 16: 1243-1247.