Journal on Data Semantics
https://doi.org/10.1007/s13740-021-00134-x

ORIGINAL ARTICLE

# DeepEx: A Robust Weak Supervision System for Knowledge Base Augmentation

Johny Moreira[1] · Luciano Barbosa[1]

Received: 12 January 2021 / Revised: 28 April 2021 / Accepted: 22 June 2021

**Abstract**
Knowledge bases allow data organization and exploration, making easier the data semantic understanding and its use by machines. Traditional strategies for knowledge base construction and augmentation have mostly relied on manual effort or automatic extraction of content from structured and semi-structured sources. In this work, we present DeepEx, a system that autonomously extracts missing attributes of entities in knowledge bases from unstructured text. We use Wikipedia as data source. Given entities on Wikipedia represented by their articles (text and infobox), DeepEx uses a classifier to detect sentences in the articles mentioning the possible missing attributes of the entities and then employs a deep-learning extraction model on those sentences to identify the attributes. The sentence classifier and attribute extractor are built with labels automatically produced by a weak supervision approach using infobox structured information as supervision source. We have compared our strategy with previous approaches to this problem on 29 different attributes from 4 domains. The results showed that our extraction pipeline achieved statistically superior performance in comparison with some baselines and variations of our approach.

**Keywords** Information extraction · Deep learning · Weak supervision · Unstructured sources · Partial matching · Wikipedia

## 1 Introduction

Due to its semantic richness, knowledge bases (KBs) or knowledge graphs (KGs) have been used for different tasks such as improving the quality of the results of web search (Halevy et al. [15]) and question-answering systems (Ferrucci et al. [13]). Since the quality of knowledge bases might have a great impact on these tasks, practitioners and researchers have been working on solutions to build and maintain them. Knowledge bases can be curated by an organization or people through crowd-sourcing, or created using automatic or semi-automatic approaches (Paulheim [28]). These strategies are though hardly complete or free of error: Knowledge bases might have wrong or missing information about attributes of entities or relations between entities.

✉ Johny Moreira
jms5@cin.ufpe.br

Luciano Barbosa
luciano@cin.ufpe.br

[1] Centro de Informática, Universidade Federal de Pernambuco, Av. Prof. Moraes Rego, 1235 - Cidade Universitária, Recife - PE 50670-901, Brazil

In this work, we are interested in the problem of knowledge base completeness, specifically, in improving the coverage of KBs with respect to missing attributes and its values for existing entities. A previous study by Min et al. [24] shows the magnitude of this problem on KBs: On the Freebase knowledge graph (Bollacker et al. [4]), for instance, 93.8% of people represented by the entity person have no known place of birth and 78.5% have no known nationality. In the work of Dong et al. [11], the numbers are even lower: 71 and 75%, respectively. The latter also states that the coverage for less common relations or attributes can be even lower.

Because of its structural nature, Wikipedia infoboxes have been widely used to create knowledge bases (Paulheim [27]) such as Google Knowledge Graph (Dong et al. [11]), DBPedia (Lehmann et al. [20]), Wikidata (Vrandečić and Krötzsch,[45]) and YAGO (Suchanek et al.[42]). However, due to automatic construction and also the constant evolving of real-world information, those KBs might miss important information about entities that can appear in the unstructured text of Wikipedia articles or regular web pages. This work proposes a pipeline for extraction of attributes of entities from unstructured text, based on machine learning models built from labels produced by a weak supervision approach. More specifically, we propose DeepEx (Deep Learning Extraction), a system that, given a Wikipedia article describing an entity, uses a classifier (Sentence Classifier) to identify the sentences in the text that mention possible missing attributes of the entity and then applies a deep-learning extraction model (Attribute Extractor) to extract the attributes from the candidate sentences, adding them to the article's infobox. Our Sentence Classifier and Attributes Extractor are supervised learning models built using weak supervision, i.e., labels are provided to train the models by an automatic process instead of manually, as in regular supervision.

Weak supervision allows to automatically label large datasets for training the models, as opposed to manual labeling, which demands a reasonable amount of human effort. On the other hand, due to its automatic nature, weak supervision can add noise to the training data. To deal with that, we propose a robust weakly supervised solution that works as follows. Given a Wikipedia article that contains both the infobox and text describing an entity, our solution uses a soft-matching algorithm, Soft TF-IDF by Cohen et al. [8], to label the tokens in the sentences that have high similarity to either the name of attributes or their values that are present in the articles' infobox. Next, a self-training approach is applied to build the Sentence Classifier. For that, the sentences with high and low matching scores are used as positive and negative examples, respectively, to train the initial Sentence Classifier. That classifier is then used to label the sentences with borderline scores as positive or negative based on their class probability: Sentences with high probability of mentioning attributes are considered as positive and the other ones as negative examples to train the final Sentence Classifier. Finally, the sentences predicted as positive by the classifier and their labeled attribute tokens are used to train the Attribute Extractor for each attribute.

Previous approaches (Lange et al. [48], Wu and Weld [19]) have also proposed weakly supervised strategies to deal with the problem of KB augmentation performing extraction on Wikipedia articles while using infoboxes as Knowledge base. Wu and Weld [48] apply rules for exact and partial matching of tokens, while Lange et al. [19] present a syntactical structure analysis of attribute values for automatic pattern definition which leads the labeling step. As opposed to them, we employ a soft-matching strategy based on Soft TF-IDF, which does not rely on rules, pattern generation or partial matching of tokens; instead, the Soft TF-IDF strategy relies on the partial matching of characters significant words in the corpus. Based on our experimental evaluation, Soft TF-IDF has showed better performance. Also, instead of using traditional sequence-based extractors as Conditional Random Fields for building the attribute extractors, as previously, we train deep neural network models to perform this task. We also investigate the use of the BERT Transformer (Devlin et al. [10]) for token classification task as the attribute extractor of our pipeline, which have not shown good performance.

We have performed an extensive experimental evaluation on 29 attributes of 4 different Wikipedia domains. The results show that our KB augmentation system[1] was able to obtain high values of F-score for most of the attributes on those domains, and outperformed the baselines in the evaluated scenarios.

The remaining of this paper is organized as follows: Sect. 2 presents basic concepts of Wikipedia data structures explored by this work. Section 3 presents the proposed extraction pipeline to augment knowledge bases from natural language text. Section 4 shows the performed experimental evaluation of the proposed extraction pipeline. Section 5 discusses the works directly related to ours. For last, Sect. 6 concludes the paper presenting the contributions from our work, as well as limitations and future lines of work.

## 2 Preliminaries

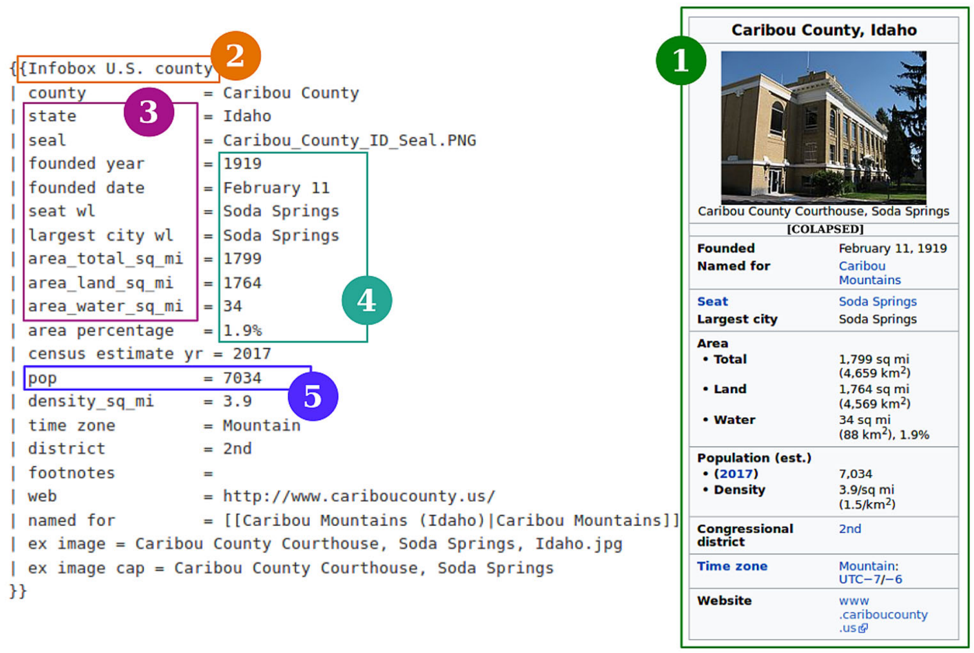
Knowledge bases are machine-readable repositories for knowledge, and Wikipedia is a large knowledge source of real-world information. Although Wikipedia presents a vast structure to organize its information (such as Categories, Subcategories, Infoboxes, and Infoboxes Templates), it is mostly designed for human consumption. In this section, we present some basic concepts and descriptions that help explain our solution for the weakly supervised knowledge base augmentation task described in Sect. 3. First we give some concepts on knowledge bases according to Balog [2]. Then are given some descriptions on Wikipedia's Infoboxes structure and definition. Figure 1 illustrates the structure of Infoboxes and the main source of information we use in our pipeline.

### 2.1 Knowledge Base

KBs are graphs in which nodes represent real-world objects and edges represent relations between them. These objects represented by nodes on graphs are also called Entities. Although the terms knowledge base (KB) and knowledge graph (KG) represent the same concept, Balog [2] highlights that

*"when the emphasis is on the relationships between entities, a knowledge base is often referred to as a knowledge graph."*

[1] The source code of our solution and datasets used in this paper are publicly available on https://github.com/guardiaum/DeepEx.



```
{{Infobox U.S. county
| county               = Caribou County
| state                = Idaho
| seal                 = Caribou_County_ID_Seal.PNG
| founded year         = 1919
| founded date         = February 11
| seat wl              = Soda Springs
| largest city wl      = Soda Springs
| area_total_sq_mi     = 1799
| area_land_sq_mi      = 1764
| area_water_sq_mi     = 34
| area percentage      = 1.9%
| census estimate yr = 2017
| pop                  = 7034
| density_sq_mi        = 3.9
| time zone            = Mountain
| district             = 2nd
| footnotes            =
| web                  = http://www.cariboucounty.us/
| named for            = [[Caribou Mountains (Idaho)|Caribou Mountains]]
| ex image = Caribou County Courthouse, Soda Springs, Idaho.jpg
| ex image cap = Caribou County Courthouse, Soda Springs
}}
```

**Caribou County, Idaho**

Caribou County Courthouse, Soda Springs

[COLAPSED]

| | |
|---|---|
| Founded | February 11, 1919 |
| Named for | Caribou Mountains |
| Seat | Soda Springs |
| Largest city | Soda Springs |
| Area | |
| • Total | 1,799 sq mi (4,659 km$^2$) |
| • Land | 1,764 sq mi (4,569 km$^2$) |
| • Water | 34 sq mi (88 km$^2$), 1.9% |
| Population (est.) | |
| • (2017) | 7,034 |
| • Density | 3.9/sq mi (1.5/km$^2$) |
| Congressional district | 2nd |
| Time zone | Mountain: UTC−7/−6 |
| Website | www.cariboucounty.us |

**Fig. 1** Wikicode (left) example to render Infobox (right) on Wikipedia page. 1—Infobox instance, rendered from Wikicode. 2—Infobox template mapping. 3—Definition of properties. 4—Assign of values to properties. 5—Property-value tuple

## 2.2 Entity

An entity is a uniquely identifiable object or thing, characterized by its name(s), type(s), attributes and relationships to other entities (Balog [2])

## 2.3 Entity Attributes

According to Balog [2] is the set of characteristics or features of an entity. Typically different types of entities are characterized by a different set of attributes. The values of attributes are always represented as literals and may also be accompanied by data type information .

## 2.4 Infoboxes

Infoboxes are a fixed-format table on Wikipedia articles that present concise and relevant information to the topic (see item 1 in Fig. 1). They are mainly composed of property-value tuples.

## 2.5 Infobox Template

The Infobox Template[2], also called infobox type or infobox class, provides standardized information across related articles, i.e., suggesting properties to be used when filling infobox information (see item 2 in Fig. 1). Although the suggested properties are not mandatory, an infobox template must be informed when creating or editing an infobox instance.

## 2.6 Wikicode

Also known as Wikitext or Wiki markup[3], Wikicode is a syntax provided by the MediaWiki foundation. This markup language is a simplified alternative to HTML and allows the online formatting of Wiki pages. In Fig. 1, we illustrate on the left side an example of Wikicode of an infobox and on the right side this infobox rendered in a Wikipedia page.

## 3 Knowledge Base Augmentation System

We present in this section our proposed system to augment knowledge bases from text: DeepEx. The architecture of DeepEx, presented in Fig. 2, is divided into three modules:

- *Data Labeling* is responsible to automatically label the data used to build the attribute extractors. It is composed of two components. The *Schema Discovery* receives as input the name of the infobox template representing a domain of interest, collects infobox instances that instantiate that template and computes the most used properties among them, which we call the domain schema. The second component, *Weak Supervision*, automatically labels tokens in the sentences on Wikipedia articles based on

[2] https://en.wikipedia.org/wiki/Help:Infobox.

[3] https://en.wikipedia.org/wiki/Help:Wikitext.

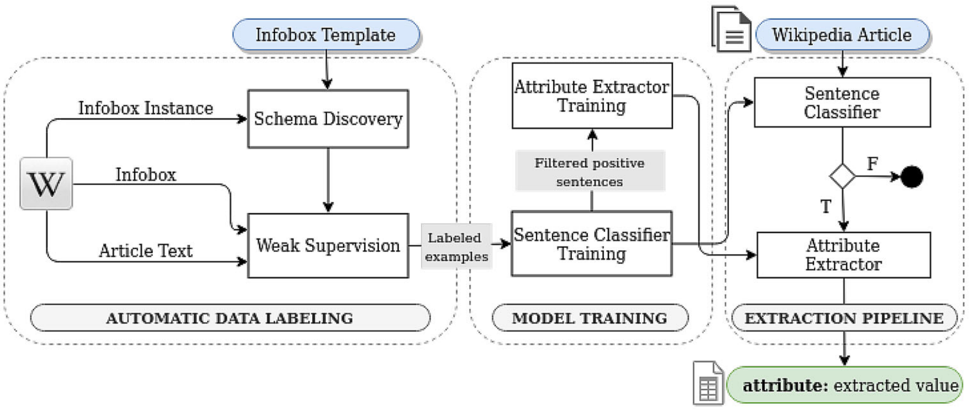


**Fig. 2** DeepEx pipeline extraction architecture

their matching with attributes in the infobox present in the domain schema.

- *Model Training* comprises the training of the sentence classifiers and attribute extractors. The *Sentence Classifier Training* receives as input the labeled sentences coming from the *Weak Supervision* and trains classifiers used to detect sentences in the Wikipedia text that mention properties in the domain schema. The *Attribute Extractor Training* uses the labeled tokens in those sentences to build the extractor that identifies the value of the missing domain schema attributes of entities in Wikipedia articles;
- *Extraction Pipeline* is responsible to process candidate articles to extract attributes of entities in the domain. the *Sentence Classifier* filters sentences from the input Wikipedia article that contain attributes of domain schema and passes them to the *Attribute Extractor* which performs the attribute extraction.

In the remaining of this section, we provide further details about each one of DeepEx's modules.

## 3.1 Data Labeling

We perform the KB augmentation only on the attributes belonging to a predefined set since the attribute extraction task requires a previously defined schema to be filled. Similar to Wu and Weld [48], we define the schema of a template by relying on the most common properties of entities that use a given template. More specifically, the component Schema Discovery receives as input the name of an infobox template, representing the domain of interest, retrieves all infobox instances using the given template, and defines the domain schema by selecting the properties that occur in at least 60% of the documents.

Next, the Weak Supervision automatically labels examples to train the models that perform the extraction of attributes in the domain schema present in the text of the Wikipedia article. For that, given a Wikipedia article representing an entity, for each property-value pair in its infobox, it tries to match it to tokens in the sentences of the article. A simple strategy to perform this task would be to do exact string matching. This approach, as explained by Takamatsu et al. [43], might be too restrictive, mostly because of the difference between the spelling of property names and values. For instance, in the strings of the property names used in this work, which come from Wikicode, there are underlines, joint words or abbreviations such as pop for population, *sq* for square, mi for miles, and yr for year. To deal with that, we apply a partial matching strategy to match property-value tokens in the infobox and sentence tokens in the article text: the Soft TF-IDF (with Jaro Winkler) (Cohen et al. [8]) similarity measure. (see Fig. 3). Given a sentence from a Wikipedia article and a property belonging to the domain schema, the algorithm calculates the Soft TF-IDF similarity between consecutive chunks of $n$ words of the sentence (sliding window) and the tokens of the property-value pair that contains the selected attribute. Note that sentence and property-value tokens can be either textual, numerical, or mixed, and for all cases, the Soft TF-IDF operates equally. The sentence containing a chunk with the highest similarity value and with a value higher than a given threshold is then considered a positive example for the sentence classifier for that property. We define three different levels of confidence for the presence of the property value in the sentence: high (similarity score equals or above 0.5); borderline (score between 0 and 0.5); and low (score equals to zero). In addition, the algorithm labels the matched tokens of the positive sentences with tags: PROP for tokens related to the attribute name, VALUE for tokens related to the attribute value or $O$ for other types of tokens. The labeled tokens are used to train the attribute extractor.

In Fig. 3, we present a concrete example of how Weak Supervision works. It calculates the Soft TF-IDF between the property "largest city wl" and its value "Soda Springs" with each 5 consecutive words of the sentence extracted from the article. For each sliding window, according to Eq. 1, a Soft

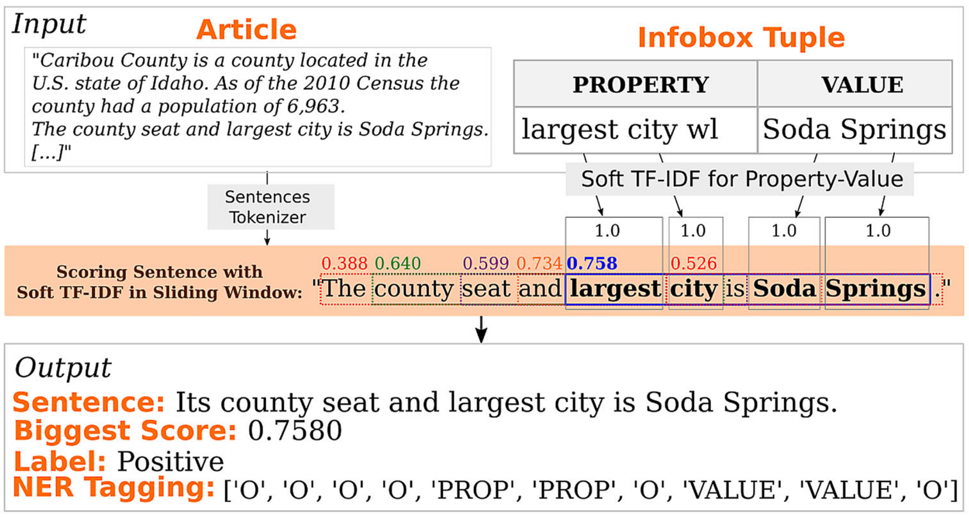

Input
Article
"Caribou County is a county located in the U.S. state of Idaho. As of the 2010 Census the county had a population of 6,963. The county seat and largest city is Soda Springs. [...]"
Infobox Tuple
PROPERTY
VALUE
largest city wl
Soda Springs
Soft TF-IDF for Property-Value
Sentences Tokenizer
1.0
1.0
1.0
1.0
Scoring Sentence with Soft TF-IDF in Sliding Window:
0.388 0.640 0.599 0.734 0.758 0.526
"The county seat and largest city is Soda Springs."
Output
Sentence: Its county seat and largest city is Soda Springs.
Biggest Score: 0.7580
Label: Positive
NER Tagging: ['O', 'O', 'O', 'O', 'PROP', 'PROP', 'O', 'VALUE', 'VALUE', 'O']

**Fig. 3** Example of sliding window for labeling sentences and NER tagging using Soft TF-IDF (with Jaro Winkler) measure

TF-IDF score is taken between the tokens in the window and the tokens in the property-value pair, this score indicates the label given to the sentence.

Since the higher Soft TF-IDF similarity inside a window of tokens is very high (0.7580 in the second last window), the sentence is chosen as a positive training example for the sentence classifier's training. The sliding window is also applied as an optimization step that prevents the similarity score to be too low, it also restricts the search space for labeling property-value tokens present in the sentence. Over the selected window with highest score another score is taken between each token in the window and each token in the property and value elements separately. The tokens with the highest score are matched, and the label is assigned according to the respective element. In Fig. 3 the tokens "largest" and "city" from the property element in the infobox tuple present similarity 1.0 to the respective tokens in the second last window of the sentence, while the "wl" does not match with any other token. In the sentence the two matched tokens are labeled as *PROP*. The same step is performed for the VALUE tokens. Unmatched tokens in the sentence are labeled as other.

$$\text{Soft}_{\text{TF}-\text{IDF}}(S,T) = \sum_{w \in \text{CLOSE}(\theta,S,T)} V(w,S) \times V(w,T) \times D(w,T) \quad (1)$$

$$V(w,C) = \frac{V'(w,C)}{\sqrt{\sum_{w'} V'(w,C)^2}} \quad (2)$$

$$V'(w,C) = log(TF_{w,C}+1) \times \log(\text{IDF}_w) \quad (3)$$

where

C: the corpus (the set of sentences in the articles and tuples in the infobox)
S: the set of tokens in the sliding window,
T: the set of tokens in the property-value tuple,
V(w, C): TF-IDF normalization, see equation (2)
V'(w, C): the TF-IDF weight of token w computed based on $C$. See equation (3).
$\text{TF}_{w,C}$: the frequency of word $w$ in $C$,
$\text{IDF}_{w,C}$: the inverse fraction of sentences in $C$ that contain $w$,
CLOSE $(\theta, \text{S}, \text{T})$: the set of words $w \in S$ such that there is some $v \in T$ such that $dist(w,v) > \theta$. In our experiments we set $\theta = 0.8$.
D (w, T): similarity measure. For $w \in \text{CLOSE}(\theta, S, T)$, let $D(w,T) = \max_{v \in T} dist(w,v)$. In our case, the similarity measure is Jaro Winkler (Cohen et al., [8]).

## 3.2 Model Training

As aforementioned, the Weak Supervision provides labeled examples to build the Sentence Classifier and the Attribute Extractor. For each attribute belonging to the schema domain, DeepEx builds a binary classifier that predicts whether the sentences in a given article mention the attribute or not. Each sentence classifier is built as follows. The sentences with high matching scores in the Weak Supervision step are considered as positive examples and the ones with low scores as negative examples to train the initial Sentence Classifier. Sentences with borderline scores are relabeled using self-training to increase the number of training examples to eventually improve the quality of the Sentence Classifier. For that, borderline sentences predicted by the initial sentence classifier with probability greater or equal to 0.9 are relabeled as positive samples and the remaining ones as negative to train the final sentence classifier. This step is applied in order to capture possible positive sentences not detected by the soft matching.

The Sentence Classifier uses bag-of-word features with TF-IDF weighting and is built using the support vector machine (SVM) algorithm  Cortes and Vapnik [9] since it has showed to be effective for text classification in comparison to other machine learning classifiers (Baharudin et al. [1]). We specifically use probabilistic SVM (Chang and Lin [6]) to obtain probability estimates useful in the self-training process previously mentioned. Since the initial datasets produced by the Weak Supervision are highly unbalanced (numbers of negative examples much higher than positive ones), which can harm the performance of the classifier and lead to high training time, we applied a random under sampling over the negative examples to build balanced training sets for the classifier: same number of positive and negative instances. Note that deep neural network architectures have already shown comparable or superior performance than SVM for text classification (Hartmann et al. [16]). However, given the number of neural network architectures to consider as well as hyper-parameters to tune, we opt by using a classic model for sentence classification.

Similar to the Sentence Classifier, the Model Training creates an extractor for each attribute in the domain schema. The tagged tokens (PROP, VALUE or *O*) of the sentences predicted as relevant for the attribute by the Sentence Classifier are used to train the Attributed Extractor. We implement it using sequence-based neural network models such as long short-term memory networks (LSTM) and convolutional neural networks (CNNs) due to their good performance on NLP tasks (Kim [49],Young et al. [17]). Concretely, we apply the hybrid deep neural network presented in Fig. 4. Given an input sentence, each of its words is represented by a concatenation of three different representations: pre-trained word embedding, character-based embedding and regular word features.

Currently, architectures for language modeling as Peters et al. [32], Radford et al. [34] and Devline et al. [10] are widely used in NLP tasks mainly for building deep contextualized word embeddings either by using LSTM networks or transformers (Vaswani et al. [44]). However, for the sake of simplicity we decided to employ into our pipeline pre-trained word embeddings. These word representations capture sentence context where words occur and, as a result, words appearing in similar contexts are set close to each other in the embedding space. As pre-trained word embeddings, we use Stanford's GloVe (Pennington et al. [31]), which showed competitive results on NLP tasks (Chiu and Nichols [7]), and is already trained on 6 billion words from Wikipedia and Web text.

The second word representation is the character-based word embeddings learned using the character-level CNN presented in Fig. 4. For that, the network first uses a lookup table composed of numbers (0–9), special characters, and upper and lower letters (a-z, A-Z) to output a character embedding. The character set also includes tokens for PADDING and *UNKNOWN*, which are used for padding the words according to the CNN window size, and to identify characters not present in the lookup table, respectively. The character-level features generated by the lookup table for each character in the word are sent to an embedding layer. A dropout (Srivastava et al. [41]) is performed before passing it to the CNN, and subsequently to a max pooling layer. The output of the CNN is passed through a new dropout layer which results in the word representation based on its characters.

Each word is also represented by the following features: allCaps (all characters in upper case), upperInitial (the first character is uppercased), lowercase (all characters are lowercase), mixedCaps (the token is composed of mixed cases), numeric (all characters are digits), containsDigit (the token contains at least one digit), mainlyNumeric (more than half of the characters are digit) and noinfo (in the case the token does not fit all other features).

The input sentence with this new word representation (concatenation of word embedding, character-based word embedding and regular features) is fed into a bidirectional LSTM model (BLSTM) (Graves and Schmidhuber [14]). BLSTM allows the information of long contexts to be saved, and at the same time performs predictions of the current state looking forward and backward in the input sequence. This is useful in our context, since Wikipedia articles can present a variety of writing styles, containing short and long sentences. The activation function of each output state of the BLSTM is a softmax function that outputs the probability distribution of its correspondent input token of belonging to one of the extraction classes: PROP, VALUE or OTHER. The class with the highest probability is assigned to the respective input token. The model architecture (see Fig. 4) is based on Chiu and Nichols [7], which has applied a similar architecture to the NER tagging task.

## 3.3 Extraction Pipeline

The Extraction Pipeline module performs the main task of our system: given the Wikipedia article of an entity in a domain, it extracts values of the attributes from the article's sentences. For that, it requires trained models (Sentence Classifiers and Attribute Extractors) for each attribute in the domain schema. The pipeline works as follows. Given a Wikipedia article, it executes a sentence tokenizer over the article text to separate its sentences. All sentences are passed to each attribute's Sentence Classifier. If the Sentence Classifier detects the existence of possible values for the respective attribute, the sentence is passed to the Attribute Extractor, otherwise the sentence is discarded. The Attribute Extractor returns the probabilities of each token in the sentence be a property,

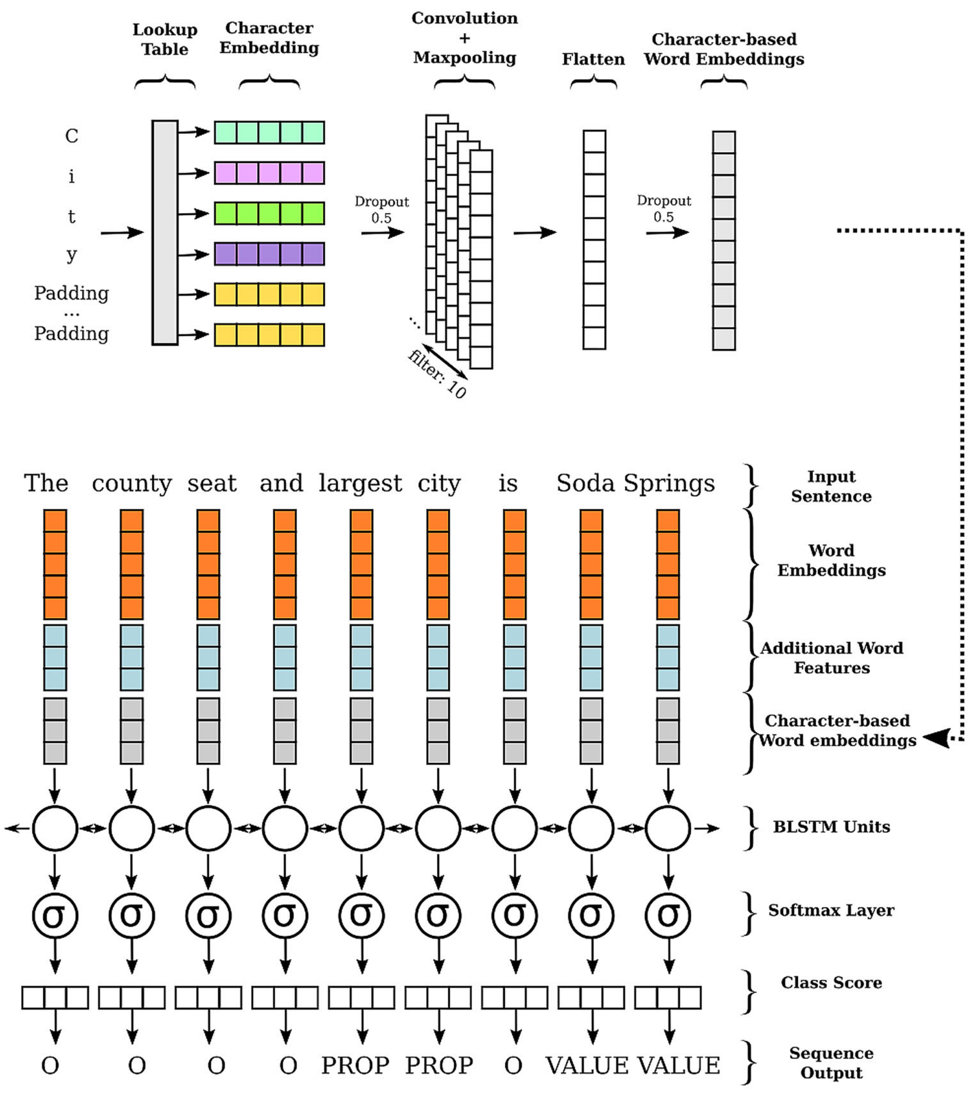


**Fig. 4** TOP—the CNN extracts character features from each word. BOTTOM—the BLSTM for tagging properties and property-values. The output is the sequence tagging

a value, or none of them (other) and considers the class with the highest probability as the predicted one. Since different candidate values for a same attribute can be detected in distinct sentences in the same article, the algorithm chooses the property value with the highest overall probability. For the cases where the tokens predicted as value appear consecutively in the sentence (e.g., ...largest city is Soda Springs - ...PROP, PROP, OTHER, VALUE, VALUE), they are joined as one value (e.g., Soda Springs, VALUE), and their probabilities are added.

## 4 Experimental Evaluation

We have performed an extensive performance evaluation of our solution on Wikipedia data by analyzing its overall performance and comparing it with previous approaches.

### 4.1 Experimental Setup

#### 4.1.1 Dataset

We built our infobox dataset using the English Wikipedia dump of October of 2016[4] and released in 2017. We used the mwparserfromhell[5] tool to parse the infoboxes in Wikicode format. Regarding the articles' text, we tried to obtain them directly from the Wikipedia dump but we found some noisy information on its annotation. Because of that, we collected them from the nif-context DBpedia[6] dataset, which contains clear plain text from articles in Wikipedia. This DBpedia dataset comes from the same Wikipedia dump we used to obtain the infoboxes. To evaluate the performance of

[4] http://wikidata.dbpedia.org/develop/datasets/dbpedia-version-2016-10.

[5] https://mwparserfromhell.readthedocs.io/en/latest/.

[6] http://downloads.dbpedia.org/2016-10/core-i18n/en/.

our work, we used 4 infobox templates: US County, Artist, Airline and University. They are among the 125 most used templates on Wikipedia and 3 of them (US County, Airline and University) were also used by previous works (Lange et al. [48], Wu and Weld [19]). The number of articles for each domain defined by its Wikipedia template is: 2957 for US County, 3852 for Airline, 13,472 for Artist and 19,363 for University. For each template, we selected 50 articles for validation and 50 for testing, and manually labeled the attributes in the sentences of those articles.

Overall, a total of 29 attributes were selected by the Schema Discovery method described in Sect. 3.1 (see Table 1): 12 on US County, 7 on Airline, 5 on Artist and 5 on University. The properties present different data types: numerical, textual, alphanumeric, and multivariate. They also vary in terms of number of tokens, number of chars and the size of the sentence where they are located as shown in Table 1. It is possible to identify that while some properties are present in short sentences, e.g., sentences presenting the seat_wl have 8.42 tokens on average, others are in large sentences, e.g., sentences presenting the attribute area have an average of 40 tokens. Also, in Table 1, one can notice the differences in the size of the values: While the majority of the properties present values with size close to 1 token, there are cases where they present 4 tokens on average, e.g., University's name, Airline's headquarters, and US County's named_for.

### 4.1.2 Approaches

We executed the following approaches for comparison:

- Kylin:
  Proposed by Wu and Weld [48] it looks for automatic creation and completeness of infoboxes. It employs strict heuristics for automatic data labeling and traditional models (maximum entropy to build sentence classifiers and CRF (Lafferty et al., [18]) as attribute extractor).
  For further details we refer the reader to Wu and Weld [48].
  An issue we had to execute this approach was to apply the Maximum Entropy model with Bagging implemented by Mallet for sentence classification, mainly because its poor documentation. Because of that, we changed the classifier model implementation for a Logistic Regression model with Bagging implemented in Scikit library[7]. For the CRF, We used its Scikit implementation: CRFSuite[8] with LBFGS (Byrd et al., [5]) feature weight estimation method.
- iPopulator: It is a rule-based system proposed by Lange et al. [19].
  It applies a fuzzy matching strategy for automatic labeling of sentence tokens. The fuzzy matching is composed of two functions, one for textual values and another for numeric values. The labels used for tagging the sentences are the position values based on the initially obtained pattern. iPopulator also applies a CRF model. We also used the CRFSuite as CRF implementation for the extraction. For further details we refer to Lange et al. [19].
- DeepEx-BLSTM+CNN: The proposed approach described in Section 3. The probabilistic SVM implementation of the Sentence Classifier was provided by the Scikit package with the default Radial Basis Function (RBF) kernel enabling the probability estimate. The attribute extraction network was trained with the *NAdam* optimizer (Dozat, [12]). The number of epochs was defined by an early stopping strategy after 5 iterations with no significant improvement over the validation set. Since we built a model for each property in the four domains and the high number of hyper-parameters in the model, we fixed values of some hyper-parameters and tuned others. The tuned hyper-parameters and respective search space for each one were convolution window [3, 5], CNN output size [10, 25, 50], and LSTM state size [100, 200, 300].
  The Character-level CNN presents two dropout layers (probability of dropout equals to 0.5), 10 filters and *tanh* activation. The BLSTM presents dropout rate of 0.5 applied to the input; and recurrent dropout of 0.25 (applied to the hidden states of recurrent units).
- DeepEx-BLSTM_WE: A variation of DeepEx-BLSTM+CNN in which we removed the CNN character-based word embeddings from the network.
- DeepEx-BLSTM: A variation of DeepEx-BLSTM+CNN without the CNN character-based word embeddings and the regular word features.
- DeepEx-CRF: To evaluate the impact of the attribute extractor in the proposed pipeline, we replaced the neural network by a CRF model. The applied CRF model used the same features described on Table 2 for training Kylin extractors. The CRF implementation was from the CRFSuite with LBFGS (Byrd et al. [5]) feature weight estimation method.
- DeepEx-BERT: To evaluate the impact of the attribute extractor in the proposed pipeline, we replaced the proposed neural network by the Bidirectional Encoder Representations from Transformers (BERT) model (Devlin et al. [10]). We have used the outputted labeled datasets from our weak supervision strategy to fine-tune the pre-trained BERT model for the token classification task. We use the bert-base-uncased model which consists of 12 Transformers blocks, 768 hidden units, and 12 self-attention heads, totalizing up to 110M parameters. We set a max sequence length to 120 tokens, fine-tuned for 3 epochs, set SEED = 1 for random numbers generator

[7] https://scikit-learn.org/.

[8] https://sklearn-crfsuite.readthedocs.io/en/latest/api.html.

**Table 1** Properties used in the evaluation for each infobox template and its type (T—Textual, M—Multivariate, N—Numeric, A—Alphanumeric), average count of tokens and characters for each property, and tokens in the sentences

| | Property | Type | Tokens | Char | Tokens in sent |
|---|---|---|---|---|---|
| U.S._county | Area_land_sq_mi | N | 1.00 | 3.32 | 40.65 |
| | Area_percentage | A | 2.00 | 4.04 | 40.65 |
| | Area_total_sq_mi | N | 1.00 | 3.36 | 40.65 |
| | Area_water_sq_mi | N | 1.00 | 2.76 | 40.65 |
| | County | T | 2.07 | 14.22 | 15.02 |
| | Density_sq_mi | N | 1.00 | 2.66 | 12.04 |
| | District | A | 1.00 | 2.78 | 23.67 |
| | Largest_city_wl | T | 1.37 | 8.26 | 13.42 |
| | Named_for | T | 3.29 | 20.67 | 23.66 |
| | Pop | N | 1.00 | 5.45 | 19.20 |
| | Seat_wl | T | 1.15 | 8.36 | 8.42 |
| | State | T | 1.20 | 8.28 | 14.71 |
| Airline | IATA | T | 1.00 | 2.17 | 16.17 |
| | ICAO | T | 1.43 | 4.86 | 11.71 |
| | Airline | T | 2.64 | 16.81 | 22.94 |
| | Callsign | T | 2.00 | 13.86 | 11.14 |
| | Fleet_size | M | 1.00 | 1.50 | 7.72 |
| | Founded | A | 1.01 | 4.11 | 20.43 |
| | Headquarters | T | 4.95 | 30.11 | 20.22 |
| | Birth_date | A | 2.82 | 11.44 | 27.13 |
| | Birth_place | T | 2.43 | 14.31 | 23.58 |
| | Field | M | 1.33 | 10.37 | 26.94 |
| | Name | T | 2.51 | 16.98 | 26.26 |
| | Nationality | T | 1.14 | 8.69 | 27.36 |
| University | City | T | 1.65 | 10.62 | 26.89 |
| | Country | T | 1.16 | 7.47 | 27.89 |
| | Established | A | 1.03 | 4.21 | 25.12 |
| | Name | T | 4.67 | 33.33 | 27.52 |
| | Type | M | 1.66 | 13.16 | 25.66 |

**Table 2** Features set used by CRF extractor

| Feature | Example | Feature | Example |
|---|---|---|---|
| First token of sentence | Hello World | Contains "_" | km_square |
| In first half of sentence | Hello World | Contains "%" | 20% |
| In second half of sentence | Hello World | Stop word | the; a; of |
| Start with capital | Hawaii | Purely numeric | 1929 |
| Single Capital | A | Number type | 1932; |
| All capital, end with period | CORP. | | 1,234; 5.6 |
| Contains at least one digit | AB3 | Part of Speech tag | |
| Made up of two digits | 99 | Token itself | |
| Made up of four digits | 1999 | NP chunking tag | |
| Contains a dollar sign | $20 | Previous tokens (window size 5) | |
| Start capital, end period | Mr. | Following tokens (window size 5) | |
| String normalization: capital to "A", lowercase to "a", digit to "1", others to "0" | | | |
| Example of String normalization: $TF-1 \Longrightarrow AA01$ | | | |

**Table 3** Macro Averages for DeepEx pipeline extraction execution. P—precision, R—recall, F—F-score

| | P | R | F | P | R | F |
|---|---|---|---|---|---|---|
| | US County | | | Airline | | |
| Kylin | 0.526 | 0.486 | 0.505 | 0.199 | 0.465 | 0.279 |
| iPopulator | 0.639 | 0.863 | 0.735 | **0.434** | 0.643 | **0.518** |
| DeepEx-BLSTM+CNN | **0.733** | 0.972 | **0.836** | 0.323 | **0.696** | 0.441 |
| DeepEx-BLSTM_WE | 0.709 | **1.000** | 0.829 | 0.312 | 0.649 | 0.421 |
| DeepEx-BLSTM | 0.707 | 0.958 | 0.814 | 0.324 | 0.633 | 0.429 |
| DeepEx-CRF | 0.693 | 0.968 | 0.808 | 0.361 | 0.443 | 0.398 |
| *DeepEx-BERT* | 0.342 | 0.970 | 0.505 | 0.239 | 0.451 | 0.312 |
| | **Artist** | | | **University** | | |
| Kylin | 0.270 | 0.400 | 0.323 | 0.438 | 0.409 | 0.423 |
| iPopulator | 0.581 | 0.681 | 0.627 | 0.684 | 0.682 | 0.683 |
| DeepEx-BLSTM+CNN | 0.735 | 0.977 | 0.839 | **0.732** | 0.952 | **0.828** |
| DeepEx-BLSTM_WE | **0.750** | 0.995 | **0.855** | 0.674 | 0.915 | 0.777 |
| DeepEx-BLSTM | 0.720 | 0.986 | 0.833 | 0.703 | 0.891 | 0.786 |
| DeepEx-CRF | 0.739 | 0.962 | 0.836 | 0.675 | 0.789 | 0.728 |
| *DeepEx-BERT* | 0.545 | **1.000** | 0.706 | 0.378 | **0.980** | 0.545 |

Bold indicates the best values for the metrics and/or highlights in the table

and BATCH_SIZE = 8. We embed the fine-tuned model into our extraction pipeline as the attribute extractor.

## 4.2 Performance Measures

To measure the performance of the approaches, we used a fuzzy matching approach. Extracted values that partially match the correct ones are considered true positive (TP). In our evaluation, a TP occurs when the Jaccard distance between the extracted value and the true one is lower or equal to 0.5. We adopted this strategy to not penalize the approaches due to exact matching. For instance, using the fuzzy matching strategy, the extracted value “Patrick R. Cleburne” is a TP for the real value “Major General Patrick R. Cleburne” because their Jaccard distance is 0.2. However, using exact matching in this example would indicate otherwise.

We measure the performance of the systems computing precision, recall, and F-score of each schema property described in Table 1. Since we are evaluating the approaches on 29 different attributes, we also measured the macro precision, recall, and F-score. For simplicity, when analyzing the performance of the models over a given infobox template, we will be referring to macro measures, otherwise, when referring to the performance of the models over separate property datasets we will be referring to regular precision, recall and F-score measures .

## 4.3 Results

We first present in Table 3 the results in the 4 domains by showing the average macro precision, recall and F-score that each approach achieved on the attributes of these domains. With respect to F-Score, four DeepEx variations , DeepEx-BERT excluded, outperformed the two baselines (Kylin and iPopulator) in 3 out of 4 domains. The only exception was on the Airline domain, in which iPopulator presented the best result. This occurred due to the poor performance of the DeepEx variations on the properties: fleet_size and IATA, as depicted in Table 4, which presents the performance of each approach on the individual attributes in each domain. Those two properties on Airline domain are represented by small strings: IATA codes have on average 2.17 characters and fleet_size values have on average 1.5 characters. This hinders the DeepEx’s Weak Supervision to accurately identify those properties in the articles’ text, labeling many negative examples as positive. In addition, few examples were labeled as positive for those attributes: 1226 examples for fleet_size and 391 for IATA, because they rarely occur in the articles.

The DeepEx variation using BERT as the extractor model have performed worse than all the other DeepEx variations, but still better or similar to Kylin, and for the Artist domain it have performed better than iPopulator. The poor performance of BERT as information extractor can be due to three points: (i) It cannot handle well the noise present on weakly supervised annotated datasets; (ii) the amount of annotated training data is not sufficient for properly fine-tune the model;

**Table 4** F-score results for DeepEx pipeline extraction execution

| F-SCORE | | DEEPEX BLSTM +CNN | BLSTM _WE | BLSTM | CRF | BERT | **BASELINE** IPOP. | KYLIN |
|---|---|---|---|---|---|---|---|---|
| US County | Area_land_sq_mi | **1.0000** | **1.0000** | 0.9474 | 0.8636 | 0.2759 | 0.6667 | 0.9899 |
| | Area_percentage | **0.9899** | **0.9899** | **0.9899** | **0.9899** | 0.0392 | 0.2712 | 0.9362 |
| | Area_total_sq_mi | **1.0000** | **1.0000** | **1.0000** | 0.9899 | 0.2143 | 0.6715 | 0.9796 |
| | Area_water_sq_mi | 0.8235 | 0.8235 | 0.8235 | 0.8095 | 0.1481 | 0.6667 | **0.9691** |
| | County | **1.0000** | **1.0000** | **1.0000** | **1.0000** | 0.9474 | **1.0000** | 0.0000 |
| | Density_sq_mi | 0.5075 | 0.3607 | 0.4375 | **0.5915** | 0.4615 | 0.1481 | 0.5075 |
| | District | 0.1379 | 0.1538 | 0.0870 | 0.1111 | 0.1429 | **0.3333** | 0.0000 |
| | Largest_city_wl | 0.3673 | 0.3462 | 0.3600 | 0.4444 | 0.2456 | **0.8235** | 0.0000 |
| | Named_for | **0.8471** | 0.8333 | 0.8235 | 0.6301 | 0.6301 | 0.6957 | 0.6842 |
| | Pop | 0.9691 | 0.9247 | 0.9583 | 0.9474 | 0.1818 | **0.9899** | 0.0000 |
| | Seat_wl | **0.9792** | 0.9583 | 0.9362 | 0.9011 | 0.8764 | 0.3448 | 0.3729 |
| | State | 0.9130 | 0.9011 | 0.9247 | 0.9691 | 0.9691 | **0.9899** | 0.2456 |
| Airline | Airline | 0.8506 | 0.8506 | 0.9247 | 0.8889 | 0.8235 | **0.9362** | 0.8372 |
| | Callsign | 0.4000 | **0.4286** | 0.2400 | 0.1538 | 0.0870 | 0.3529 | 0.1379 |
| | Fleet_size | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | **0.2667** | 0.0000 |
| | Founded | **0.8354** | 0.7368 | 0.8293 | 0.7463 | 0.6567 | 0.7302 | 0.6154 |
| | Headquarters | 0.6761 | 0.6471 | **0.6765** | **0.6765** | 0.5846 | 0.5581 | 0.0000 |
| | IATA | 0.0645 | 0.0513 | 0.0500 | 0.1818 | 0.0000 | **0.6667** | 0.2222 |
| | ICAO | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| Artist | Birth_date | 0.9670 | **0.9783** | 0.9451 | 0.9451 | 0.8471 | 0.2807 | 0.7750 |
| | Birth_place | 0.6667 | **0.7500** | 0.6857 | 0.7429 | 0.3929 | 0.4762 | 0.0000 |
| | Field | 0.7013 | 0.6667 | 0.6301 | **0.7179** | 0.6667 | 0.5333 | 0.0000 |
| | Name | 0.9583 | 0.9691 | 0.9691 | 0.9247 | 0.9130 | **0.9899** | 0.8372 |
| | Nationality | 0.8372 | 0.8506 | **0.8636** | 0.8095 | 0.5797 | 0.6372 | 0.0000 |
| Univer. | City | **0.7692** | 0.6667 | 0.6286 | 0.4444 | 0.4194 | 0.5507 | 0.0000 |
| | Country | **0.8471** | 0.8434 | 0.8605 | 0.7532 | 0.5373 | 0.6444 | 0.0000 |
| | Established | **0.9091** | 0.8736 | 0.8605 | 0.8148 | 0.3871 | 0.7333 | 0.7692 |
| | Name | 0.9583 | 0.9362 | 0.9796 | 0.9474 | 0.9130 | **1.0000** | 0.9474 |
| | Type | **0.6286** | 0.5455 | 0.5797 | 0.4923 | 0.2759 | 0.4848 | 0.0870 |
| **AVERAGE F-SCORE** | | **0.7105** | 0.6926 | 0.6900 | 0.6720 | 0.4557 | 0.6015 | 0.3763 |

Bold indicates the best values for the metrics and/or highlights in the table

or (iii) it has difficulty to predict numbers, since as already stated by Wallace et al. [46] character-level methods exhibit stronger numeracy than word- and sub-word-level methods like BERT.

We performed statistical tests to compare our proposed model (DeepEx-BLSTM+CNN) with the other ones, considering the F-score values of all attributes present in Table 4. We executed the Wilcoxon signed-rank test (Wilcoxon [47]) with a significance level of 0.05. The alternative hypothesis is that the median of DeepEx-BLSTM+CNN F-score values is greater than the median of the other approaches. The results in Table 5 show that the null hypothesis is rejected in all evaluated scenarios. This confirms that our proposed system (DeepEx-BLSTM+CNN) achieved statistically superior results than the other strategies: the two baselines (iPopulator and Kylin) and the variations of our approach (DeepEx-BLSTM_WE, DeepEx-BLSTM, DeepEx-CRF , and DeepEx-BERT).

The DL variations of DeepEx performed better for the majority of the properties. However, the DeepEx variation with CRF extractor presents an average F-score below the ones from the DL variations (see Table 4). DeepEx with the transformer variation has surpassed only the performance of Kylin.

Regarding the number of examples automatically labeled by the approaches in each domain, many properties in which DeepEx obtained a poor performance were the ones that its Weak Supervision approach was able to label only a small number of positive examples such as district (1584 examples) on the US county domain and ICAO (236) on

**Table 5** Results for the Wilcoxon signed-rank test

| Comparisons DeepEx-BLSTM+CNN vs | p-value | NullHypothesis $\alpha = 0.05$ |
|---|---|---|
| DeepEx-BLSTM_WE | 0.0109 | Reject |
| DeepEx-BLSTM | 0.02235 | Reject |
| DeepEx-CRF | 0.02982 | Reject |
| DeepEx-BERT | 0.000008 | Reject |
| iPopulator | 0.01484 | Reject |
| Kylin | 0.00003 | Reject |

the Airline domain. Looking at the poor results of DeepEx-BERT, comparing it to the other DeepEx variations, we can notice that attributes as US County's area_percentage (1308 examples) and Airline's Callsign (603) are some cases in which the number of labeled sentences could not be enough for fine-tuning BERT. Additionally, the cases of US County's area_land_sq_mi(4216) and pop (7073) indicate that BERT could not surpass the noise and struggles when running on numeric properties. Another point to note by those numbers is that the DeepEx's Weak Supervision generates unbalanced training sets. This was circumvented by the undersampling process during the sentence classifier training as aforementioned.

The performance of Kylin was harmed by its autonomous labeling strategy. Some properties of the Kylin pipeline could not leverage a sufficient number of training samples for the sentence classifiers, e.g., county, district and pop do not labeled sufficient negative training examples, and headquarters only 264 positive samples. In particular, for the property district in the domain US County, our approach extracted 1584 positive and 144,906 negative examples, whereas Kylin only 616 positive example and none negative instance.

Although iPopulator have managed to extract information for almost all properties in the domains, the results presented in Table 3 indicate a lack of recall in the extractions when compared to the DeepEx variations. This may have happened because iPopulator restricts its search for properties to only a few first paragraphs of the article, e.g., properties like US County's area and density, and University's type are rarely found in the first paragraphs of the text. Also, the labeling of property values fragment with the use of fuzzy matching has generated noisy training data since it is too broad leading to eventually a great number of false positive examples.

## 4.4 Weakly Supervised Annotation Evaluation

Given the results obtained from the pipeline execution we can assume that the weakly supervised approach could provide good annotated representations for training the extractors. However, to better analyze the quality of the generated training data we performed the following evaluation:

- We randomly select 1% of automatically annotated positive sentences from each template attribute;
- We run our weakly supervised approach for tokens annotation on each sentence;
- We manually evaluate each annotated sentence. We check if the property and value coming from the reference tuple are correctly matched by the sentence tokens;
- For each element of the tuple (property and value), we apply the criteria in Table 6 to indicate if the sentence contains a match of each element or not;

**Table 6** Criteria for manual evaluation of labeled tokens (PROP and VALUE)

| Evaluation | Criteria |
|---|---|
| Matching | text and numbers: the labeled tokens are exact matches to the expected tokens |
| Partial Matching | text: some tokens are labeled or there is some semantic meaning with the expected value (e.g., expected: pop, labeled: people, population) numbers: the labeled token is close to the expected value (e.g., expected: 209, labeled: 200) |
| No Matching | text and numbers: there is no matching between the expected value and the tagged tokens |

We select some automatically labeled training examples to illustrate how this manual evaluation was performed (see Table 7). The results of this evaluation are shown in Table 8. We present the proportion of the selected positively labeled training sentences that contain rightfully predicted tags (PROP, VALUE, or both). This metric takes into consideration the matches and the partial matches as positive tagging.

The DS in the column header stands for distant supervision, and consists in the traditional assumption of Mintz et al. [25] that states *"if two entities participate in a relation, any sentence that contain those two entities might express*

**Table 7** Evaluation examples of the automatically labeled tokens for training attribute extractors. TRUE PROP and TRUE VALUE indicate the reference values expected to be tagged by the Soft TF-IDF approach. PROP and VALUE EVAL indicate the manual evaluation of the tagged tokens

| LABELED SAMPLE | TRUE PROP | PROP EVAL. | TRUE VALUE | VALUE EVAL. |
|---|---|---|---|---|
| [As/O], [of/O], [the/O], [2010/O], [census/O],[,/O], [the/O], **[population/PROP]**, [was/O], **[13,801/VALUE]**, [./O] | pop | partial match | 13,801 | match |
| [As/O], [of/O], [the/O], [2010/O], [census/O], [,/O], [the/O], **[population/PROP]**, [was/O], **[20,662/VALUE]**, [./O] | pop | partial match | 20,216 | partial match |
| [Johnson/O], [County/O], [is/O], [a/O], [county/O], [located/O], [in/O], [the/O], [U.S./O], **[state/PROP]**, [of/O], **[Iowa/VALUE]**, [./O] | state | match | Iowa | match |
| [Batesville/O], [became/O], [the/O], [county/O], **[seat/PROP]**, [./O] | state | no match | Texas | no match |
| [The/O], [county/O], **[seat/PROP]**, [is/O], [the/O], [City/O], [of/O], [Fairfax/O], [,/O], [though/O], [because/O], [it/O], [is/O], [an/O], [independent/O], [city/O], [under/O], **[Virginia/VALUE]**, [law/O], [,/O], [the/O], [city/O], [of/O], [Fairfax/O], [is/O], [not/O], [part/O], [of/O], [Fairfax/O], [County/O], [./O] | state | no match | Virginia | match |
| [Geography/O], [According/O], [to/O], [the/O], [U.S./O], [Census/O], [Bureau/O], [,/O], [the/O], [county/O], [has/O], [a/O], [total/O], **[area/PROP]**, [of/O], [347/O], [square/O], [miles/O], [(/O], [900/O], [km2/O], [)/O], [,/O], [of/O], [which/O], [345/O], [square/O], [miles/O], [(/O], [890/O], [km2/O], [)/O], [is/O], [land/O], [and/O], [1.3/O], [square/O], [miles/O], [(/O], [3.4/O], [km2/O], [)/O], [(/O],**[0.4/VALUE]**, **[%/VALUE]**, [)/O], [is/O], [water/O], [./O] | area percentage | partial match | 0.40% | match |
| [The/O], **[per/PROP]**, [capita/O], [income/O], [for/O], [the/O], [county/O], [was/O], [$/O], [16,509/O], [./O] | area percentage | no match | 6.00% | no match |

**Table 8** Average proportion (by template class) of positively annotated sentences that contain right annotations for tokens

| | DS | SOFTTF-IDF |
|---|---|---|
| US COUNTY | 0.50 | 0.75 |
| AIRLINE | 0.40 | 0.70 |
| ARTIST | 0.08 | 0.57 |
| UNIVERSITY | 0.18 | 0.64 |

*that relation"*, i.e., both elements of the tuple (property and value) are required to be present in the sentence to define the sentence as a positive sample. Hence, it is a conjunction of the matches for the tags PROP and VALUE. We use the DS traditional approach as a baseline to benchmark with our proposed approach using Soft TF-IDF. Additionally, the annotation steps with heuristics for partial matching of tokens proposed by Wu and Weld [48], for Kylin, is a close variation of this distant supervision approach.

Wu and Weld [48] states that if the tuple value is present in several sentences *"KYLIN determines what percentage of the tokens in the attribute's name are in each sentence. If the sentence matching the highest percentage of tokens has at least 60% of these keywords, then it is selected as a positive training example."*, i.e., they look at the number of matched tokens between the entire property-value tuple and the candidate sentence. Hence, either the traditional DS approach and the Kylin baseline requires the presence of both property and value in the sentence.

The result of the evaluation shown in Table 8 indicates that our Soft TF-IDF approach annotates more training examples, since it does not require that both PROP and VALUE are present in the same candidate sentence. Additionally, this indicates that our proposed extractor can learn patterns from sentences containing only tokens tagged as PROP or as VALUE.

## 5 Literature Review

Previous approaches have tried to enhance existing knowledge bases by dealing with the issues of incorrect or incomplete information about entities, relationships, types, and literal values. Some of them have applied internal information for that, using, for instance, only the knowledge graph information itself to infer and add missing knowledge or identify erroneous pieces of information. In this direction, e.g., Nickel et al.[26] have focused on incorporating ontological knowledge to a factorization sparse vector for learning relations of the YAGO Knowledge Base. Other solutions (Paulheim and Bizer[29], Sleeman and Fini[30], Sleeman et al [39]; [40]) used statistical and machine learning approaches, as probabilistic and topic modeling, focusing on enhancing the DBpedia knowledge base.

Some other approaches have relied on the autonomous extraction of information from semi-structured web pages (Lockard et al.[21], [22]) or with human intervention Ristoski et al. [37]. Hence, the majority of recent works on information extraction for the semantic web (Martinez-Rodriguez et al. [23]) do not present an end-to-end weakly supervised pipeline with automatic labeling and classification of sentence and tokens.

Lockard et al. [21] performs information extraction on DOM trees by annotating training examples using a modification of the Distant Supervision (Mintz et al. [25]) assumption over semi-structured pages. First, the authors try to identify the subject entity and object entities in the page, then the next step is to annotate relations between the subject and object entities. In the context of semi-structured pages the subject entity is the topic entity of the page while the relation is a given field in the structure and object entities are values for that given field. The authors train a multinomial logistic regression model with the annotated data receiving as input a DOM tree and trying to predict the probability that this node tree represents a particular relation.

Lockard et al. [22] performs Open and Closed IE over semi-structured web pages by building a rich representation of textfields and the relationships between them by using a graph structure and applying a graph-based neural network approach. The graph-attention network builds contextual features that are passed to a binary classifier for OpenIE and a Multi-class classifier for ClosedIE. Although the model does not require the input of a known template to perform the extraction, the built graph structure depends on the representation of textfields in the page as well as the relationships between them, i.e., it requires semi-structured information. Hence, this model focuses on learning patterns between fields on semi-structured web pages in order to perform extraction, as opposed to our solution, which deals with attribute extraction in the unstructured text present in articles.

Ristoski et al.[37] builds two strategies for relation extraction, the first is based on crawling web documents and the second is by mining target relations from an external knowledge graph. When generating candidate examples for training the classification model the authors include a human-in-the-loop strategy for filtering the number of candidates, improve the data quality and avoid semantic drift. The classification model is a neural network using lexical and sentence level features with a softmax output layer for predicting customer-supplier relation between two organizations.

Related to our work are the ones exploring external unstructured sources for knowledge base augmentation. Banerjee and Tsioutsiouliklis [3] explores Wikipedia corpus and structures for relation extraction. It handles wrong labeling provided by Distant Supervision using DBpedia triples and text from articles in Wikipedia. It uses confidence values provided by co-occurrence statistics of dependency paths that are provided as weights to the a Multi-Encoder Model with three LSTM layers. The model encodes features from words, POS tags and dependency paths. The output of the hidden states is used to predict the relation. Another approach, proposed by Sáez and Hogan [38] derives statistics from Wikidata (Vrandeçić and Krötzsch, [45]) to create infobox instances. It ranks and prioritizes attribute-value pairs from entities, using it to create Wikipedia infoboxes. It does not use any other type of information such as manually-specified templates, training data, or unstructured information.

Although, Banerjee and Tsioutsiouliklis [3] and Sáez and Hogan [38] are close to the work performed here, the first works in reduce the wrong labels provided by DBPedia and the DS assumption, and the later work builds infoboxes from Wikidata. The work proposed here work tries to build or complete infoboxes from articles texts and using existing infoboxes as the main source of supervision.

Infoboxer is a tool based on semantic web technologies proposed by Yus et al. [50] that also deals with infoboxes' creation and update. It identifies popular properties in infobox instances on a category and uses the frequency to select them. It also identifies attribute value types from the DBpedia Ontology. However, the property values must still be manually informed by the user. When the value type for the schema property comes from a semantic class, Infoboxer looks only at the existing instances of this class on DBpedia to suggest possible values for auto-completion. Hence, Infoboxer works only as an auxiliary tool for infobox creation rather than an automatic approach for structuring information.

More similar to our approach, Kylin (Wu and Weld [48]) is a system that looks for pages with similar infoboxes, defines common attributes, create training datasets, and trains CRF extractors. It looks for autonomously creation and completeness of Infoboxes, as well as automatic link generation for identified nouns. Our system has focused on the first goal, autonomous creation or completeness of infoboxes using different strategies. Also, given the continuous updates of Wikipedia texts, pages, and structures, Kylin is obsolete to the current context mostly because of the strict heuristics applied to build training datasets.

Also focused on infobox enhancement, iPopulator (Lange et al. [19]) presents a process to extract information from plain text and populate infobox instances. Different from Kylin, iPopulator does not define the schema to be filled, it makes use of all suggested attributes from evaluated template types, and does not apply classifiers to filter irrelevant sentences. Also, iPopulator does not use the entire article text when looking for extractions, only the few first article paragraphs. As stated before, to build training datasets Kylin applies a set of heuristics, pursuing exact matches between Infobox property-values and text sentences. Nevertheless, as asserted by (Lange et al. [19]), often an attribute value and its occurrence in the text do not precisely match. Based on this assumption, iPopulator defines two different functions based on similarities measures, one for numerical values and the other for string values to detect matching from articles text tokens and attribute values.

Our work differs from Infoboxer (Yus et al. [50]), Kylin (Wu and Weld [48]) and iPopulator (Lange et al. [19]) which are systems directly related with the work proposed in this paper. Infoboxer is a tool to assist users during infobox creation and editing, while Kylin and iPopulator focus on autonomous infobox generation through identifying Entity properties and values for information extraction. Also, the work of Sáez and Hogan[38] does not apply information extraction in natural language texts as a means of infobox creation, instead it generates derivatives from Wikidata structured information.

Regarding weak supervision approaches, Ratner et al. [36] have recently proposed Snorkel, a system that combines weak supervision sources to create training data for building machine learning applications. The system is built under the data programming paradigm (Ratner et al. [35]) in which users can define labeling functions that represent weak supervision strategies or heuristics. These labeling functions are used to label subsets of the data and the "denoise" of the labeling process is solved by a generative model. Snorkel have shown superior performance in various tasks beating traditional approaches as Distant Supervision, rules and heuristics. However, this system still requires user intervention in the definition of labeling functions. The pipeline proposed here is end-to-end in the sense that no user intervention is needed in order to label data.

## 6 Conclusion and Future Work

In this work, we proposed DeepEx, a system for extracting structured information from text to augment Wikipedia's infoboxes. It identifies sentences that might mention attributes of an entity in the text of a given Wikipedia article. A deep-learning extraction model extracts the possible missing values in those sentences. To build the models, the system first runs a weak supervision strategy using Soft TF-IDF that allows partial matches between the tokens in the articles' infobox and their text, creating labels for sentence classifiers and attribute extractors. Our experimental evaluation shows that the average F-score of our approach on those datasets is significantly higher than the baselines.

As future work, we plan to investigate strategies to improve the labeling performed by the weak supervision step by applying neural networks as the work of Qu et al. [33]. For sentence labeling we intend to study the use of topic-based models and pattern correlations as in Takamatsu et al. [43], as well as word attention and property features as in Qu et al. [33]. Additionally, we intend to study the inclusion of the Snorkel (Ratner et al. [36]) labeling system for building training examples.

**Author contributions** JM, LB contributed to conceptualization; JM, L contributed to methodology; JM contributed to formal analysis and investigation; : JM contributed to software; JM contributed to writing—original draft preparation; LB contributed to writing—review and editing; Luciano Barbosa contributed to funding acquisition; LB contributed to supervision.

**Funding** This work was supported by Fundação de Amparo a Ciência e Tecnologia do Estado de Pernambuco (FACEPE) under funding grant No.IBPG-1172-1.03/16.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

**Data Availability** The datasets used in this paper are publicly available on https://cin.ufpe.br/~jms5/deepex-data.zip.

**Code Availability** The source code of our solution is publicly available on https://github.com/guardiaum/DeepEx.

## References

1. Baharudin B, Lee LH, Khan K (2010) A review of machine learning algorithms for text-documents classification. J Adv Inf Technol 1:4–20
2. Balog K (2018) Entity-oriented search. the information retrieval series. Springer International Publishing, New York
3. Banerjee S, Tsioutsiouliklis K (2018) Relation extraction using multi-encoder lstm network on a distant supervised dataset. In: IEEE 12th International Conference on Semantic Computing
4. Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J (2008) Freebase: A collaboratively created graph database for structuring human knowledge. Proceedings of the International Conference on Management of Data

5. Byrd RH, Lu P, Nocedal J, Zhu C (1995) A limited memory algorithm for bound constrained optimization. SIAM J Sci Comput 16:1190
6. Chang CC, Lin CJ (2011) Libsvm: a library for support vector machines. ACM Trans Intell Syst Technol 2(3):1–27
7. Chiu JP, Nichols E (2016) Named entity recognition with bidirectional LSTM-CNNs. Trans Assoc Comput Linguist 4:357–370
8. Cohen WW, Ravikumar P, Fienberg SE, et al. (2003) A comparison of string distance metrics for name-matching tasks. In: Proceedings of the International Conference on Information Integration on the Web, p 73–78
9. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–293
10. Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics
11. Dong X, Gabrilovich E, Heitz G, Horn W, Lao N, Murphy K, Strohmann T, Sun S, Zhang W (2014) Knowledge vault: a web-scale approach to probabilistic knowledge fusion. Proceedings of the 20th International Conference on Knowledge Discovery and Data Mining pp 601–610
12. Dozat T (2016) Incorporating nesterov momentum into adam. International Conference on Learning Representations Workshop
13. Ferrucci D, Brown E, Chu-Carroll J, Fan J, Gondek D, Kalyanpur AA, Lally A, Murdock JW, Nyberg E, Prager J, Schlaefer N, Welty C (2010) Building watson: an overview of the DeepQA project. AI Magazine 31:59–79
14. Graves A, Schmidhuber J (2005) Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw 18:602–610
15. Halevy A, Norvig P, Pereira F (2009) The unreasonable effectiveness of data. IEEE Intell Syst 24:8–12
16. Hartmann J, Huppertz J, Schamp C, Heitmann M (2019) Comparing automated text classification methods. Int J Res Market 36:20–38
17. Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceeding of the Conference on Empirical Methods in Natural Language Processing
18. Lafferty JD, McCallum A, Pereira FCN (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceeding of the 18th International Conference on Machine Learning
19. Lange D, Böhm C, Naumann F (2010) Extracting structured information from wikipedia articles to populate infoboxes. In: Proceeding of the 19th ACM International Conference on Information and Knowledge Management
20. Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, Hellmann S, Morsey M, Van Kleef P, Auer S et al (2015) Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web 6(2):167–195
21. Lockard C, Dong XL, Einolghozati A, Shiralkar P (2018) Ceres: Distantly supervised relation extraction from the semi-structured web. Proceeding VLDB Endowment
22. Lockard C, Shiralkar P, Dong XL, Hajishirzi H (2020) ZeroShotCeres: Zero-shot relation extraction from semi-structured webpages. In: Proceeding of the 58th Annual Meeting of the Association for Computational Linguistics
23. Martinez-Rodriguez JL, Hogan A, Lopez-Arevalo I (2020) Information Extraction meets the Semantic Web: A Survey, vol 11
24. Min B, Grishman R, Wan L, Wang C, Gondek D (2013) Distant supervision for relation extraction with an incomplete knowledge base. Proceeding of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies
25. Mintz M, Bills S, Snow R, Jurafsky D (2009) Distant supervision for relation extraction without labeled data. In: Proceeding of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing, vol 2
26. Nickel M, Tresp V, Kriegel HP (2012) Factorizing yago: Scalable machine learning for linked data. In: Proceeding of the 21st International Conference on World Wide Web
27. Paulheim H (2016) Knowledge graph refinement: a survey of approaches and evaluation methods. Semant Web 8:12
28. Paulheim H (2017) Data-driven joint debugging of the dbpedia mappings and ontology. Semant Web 81:404–418
29. Paulheim H, Bizer C (2013) Type inference on noisy rdf data. in the semantic web. Springer, Berlin
30. Paulheim H, Bizer C (2014) Improving the quality of linked data using statistical distributions. Int J Semant Web Inf Syst 10:63–86
31. Pennington J, Socher R, Manning C (2014) Glove: Global Vectors for Word Representation. In: Proceeding of the Conference on Empirical Methods in Natural Language Processing
32. Peters M, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. In: Proceeding of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics
33. Qu J, Ouyang D, Hua W, Ye Y, Li X (2018) Distant supervision for neural relation extraction integrated with word attention and property features. Neural Netw 100:59–69
34. Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving language understanding by generative pre-training
35. Ratner A, Sa CD, Wu S, Selsam D, Ré C (2016) Data programming: Creating large training sets, quickly. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, p 3574–3582
36. Ratner A, Bach SH, Ehrenberg H, Fries J, Wu S, Ré C (2020) Snorkel: rapid training data creation with weak supervision. VLDB J 29(2):709–730
37. Ristoski P, Gentile AL, Alba A, Gruhl D, Welch S (2020) Large-scale relation extraction from web documents and knowledge graphs with human-in-the-loop. J Web Semant 600:100546
38. Sáez T, Hogan A (2018) Automatically generating wikipedia infoboxes from wikidata. In: Companion Proceeding of the The Web Conference 2018, WWW '18, p 1823–1830
39. Sleeman J, Finin T (2013) Type prediction for efficient coreference resolution in heterogeneous semantic graphs. Proceeding of the IEEE 7th International Conferenec on Semantic Computing
40. Sleeman J, Finin T, Joshi A (2015) Topic modeling for RDF graphs. CEUR Workshop Proceeding
41. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
42. Suchanek FM, Kasneci G, Weikum G (2007) Yago: a core of semantic knowledge. Proceeding of the 16th International Conference on World Wide Web p 697–706
43. Takamatsu S, Sato I, Nakagawa H (2012) Reducing wrong labels in distant supervision for relation extraction. In: Proceeding of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers, pp 721–729
44. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser u, Polosukhin I (2017) Attention is all you need. In: Proceeding of the 31st International Conference on Neural Information Processing Systems, p 6000–6010
45. Vrandečić D, Krötzsch M (2014) Wikidata: a free collaborative knowledgebase. Commun ACM 57(10):78–85

46. Wallace E, Wang Y, Li S, Singh S, Gardner M (2019) Do NLP models know numbers? probing numeracy in embeddings. In: Proceeding of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pp 5307–5315
47. Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 55:192
48. Wu F, Weld DS (2007) Autonomously semantifying wikipedia. In: Proceeding of the 16th ACM Conference on Information and Knowledge Management, pp 41–50
49. Young T, Hazarika D, Poria S, Cambria E (2018) Recent trends in deep learning based natural language processing [Review Article]. IEEE Comput Intell Mag 13(3):55–75
50. Yus R, Mulwad V, Finin T, Mena E, et al. (2014) Infoboxer: using statistical and semantic knowledge to help create wikipedia infoboxes. In: 13th International Semantic Web Conference