Named Entity Extraction from Semi-structured Data Using Machine Learning Algorithms

Madina Mansurova[1(✉)], Vladimir Barakhnin[2,3], Yerzhan Khibatkhanuly[1], and Ilya Pastushkov[2,3]

[1] Al-Farabi Kazakh National University, Almaty 050040, Kazakhstan
mansurova.madina@gmail.com, x.erzhan@gmail.com
[2] Institute of Computational Technologies SB RAS, Novosibirsk, Russian Federation
[3] Novosibirsk State University, Novosibirsk, Russian Federation
bar@ict.nsc.ru, pas2shkov.ilya@gmail.com

Abstract. The modern society have been witnessed that intensive development of Internet technologies had followed to information explosion during last decades. This explosion had been expressing by an exponential growth of data volume among the low-quality information. This paper is designed to provide detailed information about some intellectual tools which are support decision taking by automatic knowledge extraction. In the first part of paper, we considered a preprocessing contains morphological analysis of texts. Then we had considered the model of text documents in the form of a hypergraph and implementation of the random walk method to extract semantically close word's pairs, in other words, pairs that often appears together. Result of calculations is matrix with word affinity coefficients corresponding to each other component of vocabulary vector. In the second part we describe training of neural network for linguistic constructions extraction. These ones include possible values of text named entities descriptors. The neural network enables to retrieve information on one preselected descriptor, for example, location, in the form of the final result of the name of geographical objects. In a general case, the neural network can retrieve information on several descriptors simultaneously.

Keywords: Entity extraction · Semi-structured data · Machine learning algorithms · Random walk method · Neural networks

1 Introduction

At the present, a great amount of data is being accumulated in local and global networks. Major part of this information is represented by natural language texts. Such texts are not structured, and, consequently, useful information can't be extracted and analyzed by conventional computing methods and tools. With rapidly increasing information amount, human text reading and simple substring search in large arrays of text data are obviously ineffective. The approaches for search performing on unstructured text arrays and extracting knowledge from it are becoming more and more demandable. A scientific direction of computer linguistics deals with solutions of such tasks [1].

N. T. Nguyen et al. (Eds.): ICCCI 2019, LNAI 11684, pp. 58–69, 2019.
https://doi.org/10.1007/978-3-030-28374-2_6

Computer linguistics include automatic extraction of structured data from non-structured or semi-structured data. Information extraction systems enable to collect data from separate parts of text and represent relevant information in form of selected relationships, knowledge bases and etc. With these systems one can put information in a semantically accurate form which grant further conclusions by using corresponding algorithms.

One of an information retrieval important branches is a named entity recognition. It means the search for named entities and their classification into predetermined categories, such as people's or organization's names, location, time value, and etc.

Solution of this subtask provide either determination the tonality of references to some company or product, or building relations between named entities, or accepting requests and answering it in natural language.

This work is a continued research of fact extraction from text documents [2]. The aim of this work is development of named entities extraction technology for news reports. To achieve this goal the following tasks were set up:

1. Carrying out a morphological analysis of texts.
2. Modeling of texts in the form of a hypergraph and the use of the random walk method for extraction of semantically closed words.
3. Creation of a neural network trained for finding the correspondence of concrete key words to descriptors.

2 Review of the Existing Methods for Information Extraction

Information extraction is not a new task in the field of a natural language processing. The main methods of information extraction are classified as follows:

Feature-based extraction methods. These methods are explained by the presence of a fixed set of features and the use of weights of features of the text elements being extracted. Thus, an extracted element, a vector of its features is built up. The most common in this class are Bayes probability classifiers [3] and hidden Markov models [4–6]. Extraction of an element comes to recognition of a certain text segment detected in its vicinity. Within the framework of this research, the method of extracting facts under study refers to this class. The peculiarity of the described in this work algorithm is the use of the random walk method for extraction of key words as well as the use of a neural network for placing facts on descriptors.

Kernel extraction methods. The essence of the method is in replacing the scalar product of vectors reflecting a characteristic representation of recognized elements with some function called the kernel [7]. This function is determined algorithmically and considers a more complex representation of recognized elements and their contexts describing the text segment structure. The disadvantage of this method is in the complexity of computing when calculating kernels as well as determination of the segment structure.

Sample matching methods. They use the samples and the rules of their comparison with the text fragments [8, 9]. The samples are meant to be a chain of limiters, where a chain is kind of pattern phases. This method is similar to the kernel method.

Ontology-based methods. In [10, 11] information is extracted using a semantic web based on ontology. The authors propose an algorithm developed on the ontological base of knowledge and carry out a frequency analysis for the results of text parsing of syntactical triples StanfordNLP [12].

Named entity recognition can be resolved by following methods:

Rule-Based Methods. The work of this method is as follows. Each token in the text is drawn into a feature set. Typical features are information about whether a token begins with a capital letter, punctuation, whether a token is a heading, organization or geographical location, to which part of speech it refers. Then a set of rules is extracted from the data in the following form: Contextual template → Action. A contextual pattern is a combination of conditions with respect to the features of a sequence of tokens. If the sequence of tokens satisfies the pattern, then this sequence is marked in accordance with the "Action" as a named entity. This method is used in specified works [13–15].

Hidden Markov Models. In this method, transition is used through a sequence of hidden states, and each state produces a token. All states are dependent on each other. Each transition from one state to another generates a data type, which can be a simple symbol or a multidimensional combination of features. Hidden Markov models were used to retrieve named entities in these works [16, 17].

Maximum Entropy Markov Models. In contrast to hidden Markov models, Markov models of maximum entropy directly form the likelihood of marking, based on states [18]. The advantage of this method is the ability to use a large set of considered features [19].

Conditional Random Fields. This method is closely related to Markov models of maximum entropy [20]. However, the restriction present in the previous method, where the probability of a certain label depends only on previous labels, has been removed. Conditional random fields take into account not only previous, but subsequent labels. Since a large range of considered tags affects the cost of training the model, it is possible to use a simplified version, considering only one adjacent label on each side. The use of this method can be found in the indicated works [21, 22].

3 The Approach to the Task of Information Extraction

To extract information, in the first stage it is necessary to extract a set of semantically (lose key) words. In this work semantically close ords (word combinations) are extracted on the basis of morphological analysis of texts and the random walk method.

3.1 Text Lemmatization

Text lemmatization consists of determination of the word form and assigning token characteristics and grammar descriptions to each form of the word.

Morphological information created for each word in the text consist of four "lines" or groups of markup:

1. The word form of a lexeme (shows a "dictionary entity" of a lexeme and determines the part of speech to which it refers).
2. A set of grammar features of a lexeme or characteristics of a word classifier.
3. A set of grammar features of a word form or word-transforming characteristics (for example, case of the noun, aspect of the verb, that is singular or plural).
4. Information about non-standard grammatical form, spelling distortion.

3.2 Random Walk Method

Let a text consisting of n documents be given:

$$D = (d_1, d_2, \cdots d_n) \tag{1}$$

In this work, we will consider separate sentences as a document. Let us introduce the following notation for a text vocabulary:

$$W = (w_1, w_2, \ldots w_n) \tag{2}$$

Simulation of the text being analyzed in the form of a hypergraph is one of the convenient and frequently used methods. Let us represent the text being analyzed in the form of a hypergraph HG(V, E), where V is a set of vertices and E is a set of hyper ribs. Here, $\bigcup_{e \in E} \mathrm{e} = V$. In the hypergraph, vertices $v \in V$ are words of the text and hyper ribs $e \in E$ are documents of the text.

Using $HG = R^{|V|*|E|}$ we will denote the hypergraph adjacency matrix:

$$h(v, e) = \begin{cases} 1, & if\ v \in E \\ 0, & if\ v \notin E \end{cases} \tag{3}$$

Let HG(V, E, w) be a weighted hypergraph, where $w : E \to R^+$ is the hypergraph weight.

In our case, the degree of the hypergraph vertex and that of the hyper rib are determined by the following expressions:

$$d(v) = \sum\nolimits_{\mathrm{e} \in E} w(e) h(v, e) \tag{4}$$

$$\delta(e) = \sum\nolimits_{v \in V} h(v, e) = e \tag{5}$$

In the task under consideration, the weight of the graph vertices, in other words – the weight of the text words, can be calculated according to the following formula based on TF-IDF method:

$$w(v_i)_{tf-idf} = \frac{tf(v_i)}{N_w} * \log \frac{N}{df(v_i)} \tag{6}$$

Where:

- $tf(v_i)$ is the frequency of word occurrence in a document;
- N_w – the sum of all words composing the document;
- N – the number of documents in the collection of documents D;
- $df(v_i)$ – a set of documents in D in which the words v_i occur.

Random walk is a mathematical model of the process of random changes – steps at discrete moments of time. It is supposed that the change at each step does not depend on previous steps and on time. By virtue of the analysis simplicity this model is often used on different fields. Also, this model a significant simplification of a real process. One-dimensional discrete random walk is a random process with discrete time $\{Y_n\}_{n\geq 0}$ having the form:

$$Y_n = Y_0 + \sum_{i=1}^{n} X_i \tag{7}$$

Where:

- Y_0 is the initial state;
- $X_i = \begin{cases} 1, p_i \\ -1, q_i = 1 - p_i \end{cases}, 0 < p_i < 1, i \in N$
- Y_0, X_i, i = 1, 2.. are independent random values.

One-dimensional discrete random walk is a Markov chain with whole states. Its initial distribution is given by the random value probability function X_0 and the matrix of transition probabilities has the form:

$$P = (p_{ij})_{i,j\in Z} = \begin{pmatrix} \ddots & \ddots & \ddots & & & & \\ & q_{-1} & 0 & p_{-1} & & & \\ & & q_0 & 0 & p_0 & & \\ & & & q_1 & 0 & p_1 & \\ & & & & \ddots & \ddots & \ddots \end{pmatrix} \tag{8}$$

That is:
$p_{i,i+1} \equiv P(X_{n+1} = i+1 | X_n = i) = p_i$,
$p_{i,i-1} \equiv P(X_{n+1} = i-1 | X_n = i) = q_i, i \in Z$,
$p_{i,j} \equiv P(X_{n+1} = j | X_n = i) = 0, |i-j| \neq 1$.

The sequence of discrete random values $\{X_n\}_{n\geq 0}$ is called a simple Markov chain (with discrete time), if:

$$P(X_{n+1} = i_{n+1} | X_n = i_n, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = P(X_{n+1} = i_{n+1} | X_n = i_n)$$

Thus, in the simplest case, conditional distribution of the subsequent Markov chain state depends only on the current state and does not depend on all previous states.

The range of values of random variables $\{X_n\}$ is called the space of chain states and number n is called the step number.

P(n) is called a probability matrix of transitions at n^{th} step:

$$p_{ij}(n) = P(X_{n+1} = j | X_n = i) \tag{9}$$

Vector $p = (p_1, p_2, \ldots p_n)^T$, where $p_i = P(X_0 = i)$ is Markov chain initial distribution. It is obvious that the probability matrix of transitions is stochastic, that is:

$$\sum\nolimits_j P_{ij}(n) = 1, \forall n \in N \tag{10}$$

Markov chain is called homogeneous if the probability matrix of transitions does not depend on the step number, that is

$$P_{ij}(n) = P_{ij}, \forall n \in N \tag{11}$$

To range the hypergraph vertices, we will generalize the random walk process to a hypergraph. Transitions between vertices in a graph take place in the process of random walk, that is transition from the given vertex to a neighboring vertex via every step of discrete time t. We can consider vertices as a set of states $\{s_1, s_2, \ldots s_n\}$ and transitions – as a final Markov chain in these states. The transition probability is computed in the form $P(u, v) = Prob(s_{t+1} = v | s_t = u)$. This means that Markov chain at vertex v will be in time t + 1 and u – in time t. in our case, Markov chain is homogeneous, the probability of transition does not depend on time t. for each vertex $\sum_v P(u, v) = 1$.

M is homogeneous, with possibilities calculated for a single transition. For all steps, the transition matrix can be computed $P \in \mathbb{R}^{|V| \times |V|}$. Transition matrix P completely embraces transitions between vertices, which show the change of a surfing movement in a random order between vertices with such probability.

In a single graph, a random walk process is clear, it is only necessary to choose a rib ascending to the target vertex with a certain probability. Nevertheless, the hypergraph in this situation somewhat differs according to structure difference. For example, in a hypergraph there may be more than two points of the vertex for a hyper rib $\delta(e) \geq 2$.

To generalize the random walk process to a hypergraph, we model a transition between two vertices incident in regard to each other in a hyper rib in the form of walk. As a whole, the random walk process is not a one-step process but a two-step one: firstly, a random surfer chooses a hyper rib e incident to the current vertex u. Secondly, in the chosen hypergraph, the surfer chooses a target v satisfying the condition $u, v \in e$.

Random walk in a hypergraph is called generalization with a special case for random walk in a usual graph. This is the presence of only one vertex in a rib, and in a hypergraph we can choose from a set of vertices. If we determine the random walk process in a hypergraph with the help of Markov chain, here a set of vertices will make up a set of states. With every time step t the surfer changes the place on the incident hyper rib for another vertex.

Let us give a general definition of random walks in the measured hypergraph taking into account the weight of vertices with hyper ribs. In this case, the random walk process will be widened using the weight of vertices and hyper ribs. The weight of vertices is determined with the help of all incident hyper ribs, this is a vector of peculiarities:

$$\vec{v}_w = \left\{\omega(v_{e1}), \omega(v_{e2}), \ldots \omega\left(v_{d(v)}\right)\right\} \tag{12}$$

That is, for each hyper rib e with vertex u the weights of vertices are different. The predicted random walk process can be described as follows: starting with vertex u the surfer chooses hyper rib e which is incident to vertex u and proportional to the hyper rib weight w(e). After that, the surfer chooses, in the same way, vertex v proportional to the vertex weight that is the considered current hyper rib.

Let us determine the incident matrix $H_w \in \mathbb{R}^{|V|\times|E|}$ in the following way:

$$h_w(v,e) = \begin{cases} w(v_b), \text{ if } v \in e \\ 0, \text{ if } v \notin e \end{cases} \tag{13}$$

Thus, if we determine the vertex degree again:

$$\delta(v) = \sum\nolimits_{e\in E} h_w(v,e) \tag{14}$$

And the hyper rib degree is:

$$d(e) = \sum\nolimits_{v\in V} (v,e) = |e| \tag{15}$$

Using the above mentioned formulas, it is possible to calculate transition matrix determination:

$$P(u,v) = \sum\nolimits_{e\in E} w(e) \frac{h(u,e)}{\sum_{\hat{e}\in E(u)} w(\hat{e})} \frac{h_w(v,e)}{\sum_{\hat{v}\in e} h_w(\hat{v},e)} \tag{16}$$

Or the matrix determination:

$$P = D_v^{-1} H W_e D_{ve}^{-1} H_w^T \tag{17}$$

Here,

- D_v – a diagonal matrix of the weighted vertex degree;
- H – an incidental matrix of hypergraph vertices;
- W_e – a diagonal matrix of hyper rib weights;
- D_{ve} – a diagonal matrix of the weighted hyper rib degree;
- H_w – an incidental matrix of the measured graph.

Where the transition matrix P is stochastic and every sum of rows is equal to 1 [23].

After computing the transition matrix P, there arises the necessity to understand random walk stationary distribution π. Stationary distribution can be calculated beginning with vector $\overrightarrow{v_0} \in R^{|v|\times 1}$. These are probabilities 1/|V| where their sum is equal to 1. Firstly, the transition matrix P^T is computed by multiplication of a vector-column $\overrightarrow{v_0}$ by $\overrightarrow{v_0} = P^T * \vec{v}$. Thus, we will go on with iteration till vector $\vec{v}$ stops changing. Multiplication of transition matrix by the vector of probability distribution gives the next step of distribution $\vec{x} = P^T * \vec{v}$. Let x be the probability of being at vertex i. Then $x_i = \sum_j p_{ij} v_j$, v_j is the probability of untimely being of the surfer on the node j and p_{ij} is the probability of transition from j to i.

If random walk is ergodic, the vector distribution probability stops changing after n steps.

3.3 Creation and Training of a Neural Network

The next task is creation of a neural network trained to extract linguistic constructions which include the possible meanings of attributes of named entities of the texts being processed. Collecting of linguistic constructions and sets of semantically close words are beginning of a neural network training by providing descriptors candidates to use it as features.

The created neural network enable to extract information on one pre-selected descriptor, for example, location, presenting, names of geographical objects as a final result. In general case, the neural network can extract information across multiple descriptors simultaneously.

For training a neural network, a training set consisting of a feature vector was constructed. For one descriptor, a feature vector was built as follows: we took a window of five words before the entry of the element of interest in the text of the article and a window of two words after it. For each descriptor, a dictionary is formed which answers for the presence of the pointed out word in it. All the features of each descriptor are collected in one "bag of words" and a feature vector is built.

A neural network is trained by showing every input dataset and subsequent error propagation. Neural network training algorithm is based on the error back propagation method. This method is popular and widespread in machine learning, so there is its general description. The "backwards" means that calculation of the gradient proceeds backwards through the network, with the layer's gradients being calculated in reverse order. Partial computations of the gradient from one layer are reused in the computation of the gradient for the previous layer. This backwards flow of the error information allows for efficient computation of the gradient at each layer versus the naive approach of calculating the gradient of each layer separately. We use a simple neural network with two hidden layers to simplify results interpretation [24].

4 The Results of Computing Experiments

To store the data, NoSQL data base of MongoDB 3 is used. The data set includes 9723 records collected from Kazakhstani portals and Kazakhstan emergency news portals. The data are marked up as follows: class 1- locality, 0 - otherwise.

An example of a record: "main functional responsibilities maintaining state records of natural and man-made emergencies which took place on the territory of the city Almaty jointly with state and local executive bodies according to the civil protection law of Republic of Kazakhstan (RK) preparation of analytical materials on areas of activity in the field of civil protection maintenance of information reference cards maintenance and operation of information systems by the state system of civil protection control and supervision in the field of civil protection administrative practice".

At first, data pre-processing was performed to avoid markup mistakes in train and test sets. For markup, a list of localities in RK was taken and using a fuzzy search algorithm (difflib library) and a search algorithm for full match normalized words (with the help of library pymorphy2 [25]) the texts with and without names were found. The texts with names presence confirmed or declined by both algorithms were in the training and test sets with corresponding marks.

Then, using the library TensorFlow we calculated TF-IDF matrices which were used by the random walk method. At the output, we had a matrix with word proximity ratios and a dictionary that matched the vector component with the word. All the texts were encoded by the values of the obtained matrix and each was marked up as 1 or 0 depending on the presence of the locality name in the text.

The experimental data were divided in the ratio 80%–20%, 20% - for the final test. And 80% were further divided correspondingly to 70%–30%, 70% - for learning and 30% - for correct validation. Validation helps determine the retrained model.

All the experiments were carried out on a neural network with two hidden layers. The first hidden layer consists of 32 neurons, the second layer consists of 10 neurons. As an activation function for neurons of the hidden layer, we chose different variations of relu and tanh, the output layer being sigmoid. We had used the default RMSE, root mean square error as loss function.

Then, a graph of RMSE versus learning curve is presented. After the 500th era the value of loss function makes up ~ 0.17, ~ 0.22, ~ 0.87 corresponding for SGD, RMSPROP, Adam optimizers, which are an acceptable result for such a simple architecture of the network. We use RMSE because as the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable. Lower values of RMSE indicate better fit (Fig. 1). RMSE is a good measure of how accurately the model predicts the response, and it is the most important criterion for fit if the main purpose of the model is prediction. According to these results we have a neural network that predict is word sequence or word a location name or not with pretty precise accuracy with SGD optimizer. It means that with our algorithm will detect the Almaty in any news report with probability about 85%.

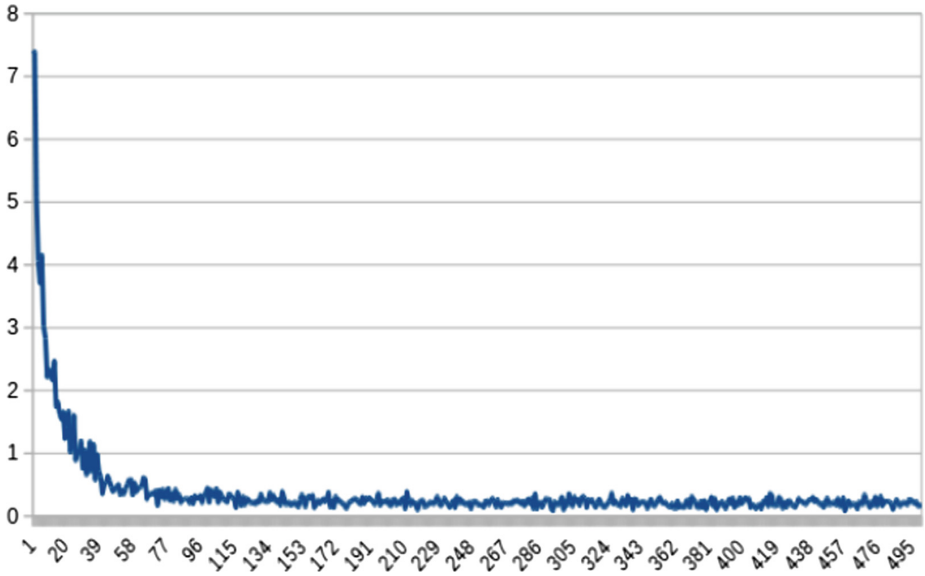

Fig. 1. RMSE loss versus learning curve

5 Conclusion

This work resulted in creation of the algorithm which allows to extract key words (word combinations) from the text corpus of uniform subject with the aim of further using the extracted key words as possible meanings of entity attributes described in the domain ontology being created assigned for organization of factual search in the widened corpus of texts of the corresponding domain. As the result we have a pipeline contains morphological parser text parts markup of speech followed by the use of the random walk method for extraction of semantically close key words (word combinations) so we can consider that task has been solved successfully. A trained neural network with a hidden layer is applied to the set of these word combinations with the aim to match a concrete word combination to a definite attribute of the entity described in the text. Thus, using a set of semantically close pairs of words, one can build an ontology, formed during the neural network operation, for a concrete document.

Acknowledgments. This work was supported in part under grant of Foundation of Ministry of Education and Science of the Republic of Kazakhstan "Development of a system for knowledge extraction from heterogeneous data sources to improve the quality of decision-making" (2018–2020) and Russian Federation RFBR grant № 18-07-01457.

References

1. Shokin, Yu.I., Fedotov, A.M., Barakhnin, V.B.: Problems of information retrieval. Novosibirsk: Sci. 196 p. (2010). (In Russian)
2. Barakhnin, V.B., Fedotov, A.M.: Building a factual search model. Vestnik NSU. Series: Information Technology, vol. 11, no. 4. pp. 16–27 (2013)
3. Pedersen, T.: A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation. ACM (2000)
4. Borkar, V., Sarawahi, S.: Automatic segmentation of text into structured records. ACM (2001)
5. Agichtein, E., Ganti, V.: Mining reference tables for automatic text segmentation. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, USA (2004)
6. Seymore, K., McCallum, A., Rosenfeld, R.: Learning hidden Markov model structure for information extraction. In: Papers from the AAAI-99 Workshop on Machine Learning for Information Extraction, pp. 37–42 (1999)
7. Zelenko, D., Aone, C.: Kernel methods for relation extraction. J. Mach. Learn. Res. **3**, 1083–1106 (2003)
8. Califf, M., Moony, R.J.: Bottom-up relational learning of matching rules for information extraction. J. Mach. Learn. Res. **4**, 177–210 (2003)
9. Dejean, H.: Learning rules and their exceptions. J. Mach. Learn. Res. **2**, 669–693 (2002)
10. Aung, A., Thwal, M.P.: Onthology based hotel information extraction from unstructured text. In: International Conference on Advances in Engineering and Technology (ICAET 2014), 29–30 March, Singapore (2014)
11. Anantharangachar, R., Ramani, S., Rajagopalan, S.: Ontology guided information extraction from unstructured text. Int. J. Web Semant. Technol. (IJWesT) **4**(1), 19–36 (2013)
12. Stanford CoreNLP: A Suite of Core NLP Tools. (2015). http://nlp.stanford.edu/software/corenlp.shtml
13. Atzmueller, M., Kluegl, P.: Rule-based information extraction for structured data acquisition using TextMarker. In: LWA (2008)
14. Chiticariu, L., Krishnamurthy, R., Li, Y., Raghavan, S., Reiss, F., Vaithyanathan, S.: SystemT: an algebraic approach to declarative information extraction. In: ACL (2010)
15. Kluegl, P., Atzmueller, M., Puppe, F.: TextMarker: a tool for rule-based information extraction. In: UIMA@GSCL Workshop, pp. 233–240 (2009)
16. Chopra, D., Joshi, N., Mathur, I.: Named entity recognition in Hindi using hidden Markov model. In: 2016 Second International Conference on Computational Intelligence and Communication Technology (CICT), pp. 581–586. IEEE (2016)
17. Malik, M.K., Sarwar, S.M.: Urdu named entity recognition system using hidden Markov model. Pak. J. Eng. & Appl. Sci. **21**,15–22 (2017)
18. McCallum, A., Freitag, D., Pereira, F.C.N.: Maximum entropy markov models for information extraction and segmentation. In: Icml 2000, vol. 17, pp. 591–598 (2000)
19. Ahmed, I., Sathyaraj, R.: Named entity recognition by using maximum entropy. Int. J. Database Theory Appl. **8**(2), 43–50 (2015)
20. Lafferty, J., McCallum, A., Pereira, F.C.N.: Conditional random fields: probabilistic models for segmenting and labeling sequence data (2001)
21. Lee, C., et al.: Fine-grained named entity recognition using conditional random fields for question answering. In: Ng, H.T., Leong, M.-K., Kan, M.-Y., Ji, D. (eds.) AIRS 2006. LNCS, vol. 4182, pp. 581–587. Springer, Heidelberg (2006). https://doi.org/10.1007/11880592_49

22. Chen, W., Zhang, Y., Isahara, H.: Chinese named entity recognition with conditional random fields. In: Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing, pp. 118–121 (2006)
23. Bellaachia, A., Al Dhelaan, M.: HGRANK: a hypergraph based keyphrase extraction for short documents in dynamic genre (2014)
24. Haykin, S.: Neural Networks and Learning Machines, 3rd edn. Pearson, London (2009). 936 p
25. Korobov, M.: Morphological analyzer and generator for russian and ukrainian languages. In: Khachay, M.Y., Konstantinova, N., Panchenko, A., Ignatov, D.I., Labunets, V.G. (eds.) AIST 2015. CCIS, vol. 542, pp. 320–332. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26123-2_31