Abjad numerals recognition in medieval arabic mathematical texts

Hadj Mohammed Djamel[1,*,†], Nacéra Bensaou[1,†]

[1]USTHB University, Laboratory for research in artificial intelligence (LRIA), BP32 EL ALIA, BAB EZZOUAR, ALGER, ALGERIE

Abstract

Abjad numerals, also called *hisāb al-jumal*, is a numeral system based on the twenty eight letters of Arabic but not in the dictionary order. In ancient Arabic mathematics, all problems and solutions sentences were completely expressed in natural language with no mathematical symbolism. The present paper is the first attempt to automatically analyze and recognize Abjad numerals in medieval Arabic mathematical texts. Since that *hisāb al-jumal* system has no ambiguity, we also translate Abjad numeral written in natural language to modern numeral system. We construct a new dataset named Hj-Tagged corpus to facilitate our study. According to the experimental results, the proposed method is efficient for automatically analyze and recognize Abjad numerals and mathematical components (such as numerical constants, Abjad numbers, mathematical operations,.. etc). We also translate Abjad terms detected in the previous step to modern numeral system, where it achieves an F1 score of 98.1%.

1. Introduction

In several medieval Arabic manuscripts such as mathematical, geographical, and astronomical texts [1], the numbers are written in a system of Arabic alpha-numerical notation. In this system each letter from the 28 Arabic letters has a specific numerical value known as the *'Adad* of that letter, and the value of a word is the sum of values to each letter compose that word. The system was known as *hisāb al-jumal* (حساب الجمل) which meant "numerological calculations" and also sometimes as *Abjad* (الأبجد) which is an acronym referring to 'alif(أ), Baã'(ب), Jïm(ج), dãl(د), the first four letters in the *Abjad* order [2].

The numbers from 1 to 9 were represented by the 9 first letters, 'alif(أ), is used to represent 1; the second letter, Baã'(ب), is used to represent 2, etc. Then the numbers 10, 20, 30, .., 90 by the next nine letters (10 = yã'(ي), 20 = kãf(ك), 30 = lãm(لا), etc), then 100, 200, 300, .., 1000 by the next letters (100 = qãf(ق), 200 = rã'(ر), 300 = syïn(ش), etc), see Table 1 for more details.

VIPERC2022: 1st International Virtual Conference on Visual Pattern Extraction and Recognition for Cultural Heritage Understanding, 12 September 2022

*Corresponding author.

†These authors contributed equally.

hadjmohd@mcmaster.ca (H. M. Djamel); i.tiddi@vu.nl (N. Bensaou)

0000-0001-6301-5152 (H. M. Djamel)

Letters & its Numerical Values					
ا ('Alif)	1	ي (yā')	10	ق (qāf)	100
ب (Bā')	2	ك (kāf)	20	ر (rā')	200
ج (JÏm)	3	ل (lām)	30	ش (syïn)	300
د (dāl)	4	م (mïm)	40	ت (Tā')	400
ه (hā')	5	ن (nün)	50	ث (Tsā')	500
و (waw)	6	س (sïn)	60	خ (khā')	600
ز (zai)	7	ع ('ain)	70	ذ (dzāl)	700
ح (Hā')	8	ف (fā')	80	ض (dhād)	800
ط (thā')	9	ص (shād)	90	ظ (zhā')	900
				غ (ghain)	1000

Table 1
The 28 letters of the Arabic alphabet are assigned numerical values (base on Abjad value order).

Hisab al-jumal numbers were used for all mathematical purposes also used for the creation of chronograms, which "consist of grouping into one meaningful and characteristic word or short phrase letters whose numerical values when totaled give the year of a past or future event"[3]. For example, a poet used it in talking about the rules of Tajweed. He made a poem stating the rules of the Arabic letters, and in the end of the poem he said: تاريخها بشرى لمن يتقنها [*the date of this poem is a good tiding to the one who masters it*] when calculated in *hisāb al-jumal* system (see Table 2), gave the year he authored that book, which was year 1198 (512+120+566) AH.

When a number is written in *hisāb al-jumal* notation, it becomes difficult to recognize it as number, especially if the *hisāb al-jumal* word make sense. For example, the equation: "اردنا أن نضرب لو دقيقة في كاح ثانية" [*We wanted to multiply law minutes by kah seconds*] (Miftāh al Hissāb [4]), the *hisāb al-jumal* numbers has a unique conversions into ordinary decimal notation, yielding "اردنا أن نضرب 36 دقيقة في 29 ثانية"

In this work, we explore the use of a Bi-directional Long Short Term Memory (BI-LSTM) network with a conditional random field (CRF) layer to automatically analyze and recognize Abjad numerals in mathematical expression in medieval Arabic mathematical texts. Additionally, we also translate *hisāb al-jumal* terms detected in the previous step which written in natural language to modern numeral system (such as decimal numbers).

In the past few years, recurrent neural networks (RNN) [5, 6], together with its variants (such LSTM and gated recurrent unit (GRU)) are generally becoming more widely known and one of the most common techniques of the natural language processing, such as part-of-speech (POS) tagging [7, 8] and named entity recognition (NER) [9, 10]. Recently, [11] applies RNN approach using Bi-LSTM with CRF for the automatic detection of words and character level features for the task of drug NER. Similarly, [12] have combined the output of a Bi-LSTM and

يتقنها	لهن	بشرى	
10 + 400 + 100 + 50 + 5 + 1	30 + 40 + 50	2 + 300 + 200 + 10	Hisab al-Jummal calculation
566	120	512	Sum total

Table 2: *Hisab al-jummal* is a calculation that assigns every letter in Arabic a specific numerical value.

a CRF as input to an Support Vector Machine (SVM) classifier for disease name recognition. For sequence tagging tasks, [13] proposed a variant of Bi-LSTM with one CRF. The paper [14] shows that a combination of Bi-LSTM with CRF and external word embeddings model achieves impressive results for Russian NER task. [15] adapted a Rule-based machine translation system using Dictionary Approach (DA) to automatically generate modern (symbolic) mathematical equations from natural language in medieval Arabic Algebra. In this paper, we propose a novel approach for automatically recognizing and translating Abjad numeral in medieval Arabic mathematical texts to modern numeral system.

Following this introduction, the remainder of this paper is organized as follows. Section 2 explains the LSTM networks, Bi-LSTM networks, and Bi-LSTM with CRF networks. Section 3 describes how to translate *hisāb al-jumal* to modern numeral system. Section 4 shows the experimental setup such as dataset construction, model architecture, and the training process. Finally section 5 summarizes our methods, results, and discusses the future work.

2. Bi-LSTM-CRF Model

Recurrent neural networks (RNNs) have proved to be efficient to learn sequential data including language model [17, 18] and natural language process [19, 20]. An RNN is a neural network that consists of an input layer x, hidden layer h and output layer y. For instance, given a sentence $x = (x_1, ..., x_n)$, an RNN uses a hidden state representation $h = (h_1, ..., h_n)$ so that it can map the input x to the output sequence $y = (y_1, ..., y_n)$.

However, standard RNNs suffer from both exploding and vanishing gradients problems [21]. On the other hand, the RNNs with the gating units such as LSTM-RNN [22] are the most

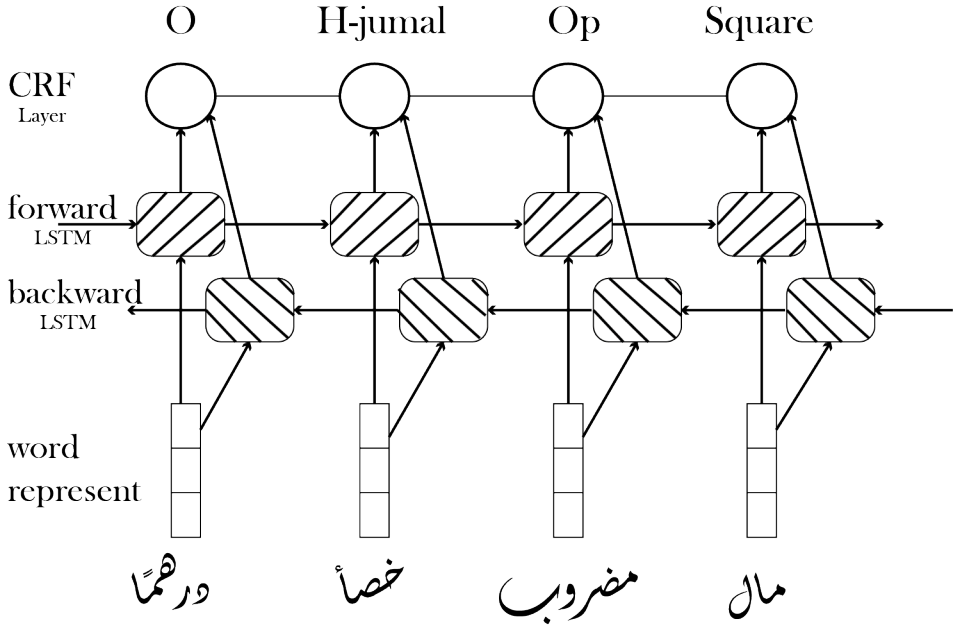

Figure 1: A bidirectional LSTM-CRF network for sequence tagging.

effective sequence models in practical applications by adding extra memory cell inherent in RNNs.

The LSTM cell can be described mathematically with the following six fundamental operational stages:

- Input Gate: $i_t = \sigma(W^{(i)}x_t + U^i h_{t-1})$
- Forget Gate: $f_t = \sigma(W^{(f)}x_t + U^f h_{t-1})$
- Output/Exposure Gate: $o_t = \sigma(W^{(o)}x_t + U^o h_{t-1})$
- New memory cell: $\tilde{c}_t = \tanh(W^{(c)}x_t + U^c h_{t-1})$
- Final memory cell: $c_t = f_t \odot \tilde{c}_{t-1} + i_t \odot \tilde{c}_t$
- Final hidden state: $h_t = o_t \odot \tanh(c_t)$

where x_t is the input vector at time t, and h_t denote the hidden state vector storing all the useful information at (and before) time t. The U and W terms denote weight matrices for each gate. The symbol σ represents the Sigmoid activation function, $\odot$ is the element wise multiplication.

In this paper, we propose to apply Bi-LSTM neural network [23] instead of a single forward network. In doing so, we can efficiently make use of past features (via forward states) and future features (via backward states) for a specific time frame. Finally, we construct our neural network model by feeding the output vectors of Bi-LSTM into a Conditional Random Field (CRF) layer [24] to jointly decode the best sequence of tags. Consider an input sentence $X = \{x_0, x_1, .., x_n\}$

and $y = \{y_1, y_2, ..., y_n\}$ is the corresponding sequence of tags for sentence X. We consider P to be the matrix of scores output by the Bi-LSTM network. P is of size $n \times k$, where k is the number of distinct tags, $P_{i,j}$ is transition probability which represents the score of the j^{th} tag of the word i^{th}, its score defined with the following form [25]:

$$s(X, y) = \sum_{n=0}^{n} A_{y_i, y_{i+1}} + \sum_{n=1}^{n} P_{i, y_i}$$

where A is a matrix of transition scores such that $A_{i,j}$ represents the score of a transition from the tag *i* to tag *j*. We use y_0 and y_n are the start and end tags of a sentence, then we add to the set of possible tags. A is therefore a square matrix of size $k + 2$.

$Y(x)$ denotes the set of possible sequence of tags for x. The probabilistic model for sequence CRF defines a family of conditional probability $p(y|X)$ with all possible sequence of tags y under the given x with the following form:

$$p(y|X) = \frac{e^{s(X,y)}}{\sum_{\tilde{y} \in Y(x)} e^{s(X,\tilde{y})}}$$

During the training, $\log$ probability of correct tag sequence $\log p(y|x)$ is maximized.

Figure 1 illustrates the main architecture of our neural network model for medieval mathematical entity recognition system in which each word is tagged with *other* (*O*) or one of six entity types: *hisāb al-jumal* (*H-jumal*), *root* (*Root*), *square* (*Square*), *cube* (*Cube*), *equal* (*Equal*), and *operation* (*Op*). The sentence of "مال و جج أجذاره يعدل س درهما" [*A square and hājj roots are equal to thirty sïn dirhems*], is tagged as *{Square Op H-jumal Root Equal H-jumal O}*.

3. Equations and Hisab Al-Jummal Calculation in Medieval Arabic Algebra

In the following, we translate some of the Arabic basic mathematical terms and notations used throughout medieval period into modern symbols:

- *Shay'* ("شيء") or *jidhr* ("جذر"), refer to unknown value(x).
- *Māl* ("مال") and *ka'b* ("كعب") represent respectively x^2 and x^3.
- Powers greater than or equal to four can be formed by combining the two words *māl* and *ka'b*. For example, *māl māl* (x^4), *māl ka'b* (x^5), and so on.
- *Dirhams* ("درهماً") or *mina al'adad* ("من العدد") represent a simple number.
- The verb *'ādala* ("عادل") is used to indicate equality ("=") in an equation.
- The one-letter word *wa* ("و") take the meaning of the modern addition ("+") depend on the context.

- Hisab al-jummal system is often used to describe numbers during the medieval Arabic period.

The numerological calculation of *hisāb al-jumal* terms requires a dictionary approach to relate every letter in the Arabic alphabet to its equivalence in a number format (see Table 1). A dictionary approach is necessary to recognize each letter and its numerical value as shown in the table below.

Let S be the sequence of n words $\{w_1, w_2, .., w_n\}$, to capture a correspondence between the word w and its numerical value t, we define an alignment p to be a set of pairs (w, t), where w is a token in S and t is sum total of w letters value. For example, consider the following sentence S: Mālan wa kāb‘ ‘ashyā‘ ta’dilu nüd‘ dirhaman (“مالان و كب أشياء تعدل ند درهماً”) [*Two māls and kab things equals nad dirhams*], given the above definitions, and knowing that the terms kab ("كب") and nad ("ند") are *hisāb al-jumal*, once t is calculated over all *hisāb al-jumal* words {t_1=(ند, 50+4) ,t_2=(كب, 20+2)}, S can be written as “مالان و 22 أشياء تعدل 54 درهماً” [*Two māls and twenty-two things equals fifty-four dirhams*]. We have shown that the passage from a sequence of words of any length to its numerical value notation is quite easy. However, no ambiguity is possible because there are exactly one unique translation of w_i to t_i.

Consider the previous example sentence (see section 2):

"مال و جج أجذاره يعدل س درهما".

Which was tagged with: *{Square Op H-jumal Root Equal H-jumal O}*. By applying the numerological calculation of *hisāb al-jumal* to the words tagged as *H-jumal*, this sentence is transformed into "مال و 11 أجذاره يعدل 60 درهما".

4. Experiments

In this section we first present our proposed architecture, shown in Figure 2, for automatically recognize and translate *hisāb al-jumal* entity in medieval algebraic equations and expressions. Next, we will discuss the construction of a new Hj-Tagged Corpus and the training detail followed by their results.

4.1. Dataset Construction and Evaluation

We evaluate our proposed system on Hj-Tagged Corpus, constructed from the AMAK Dataset [15] which consists of medieval-modern equations pairs. We implement a simple dictionary-based method to detect and replace all numbers in the collected medieval equations (the numbers are referred to as num token) with a random numerical entity in the *hisāb al-jumal* system. We also added several algebraic expressions which has numbers already written in *hisāb al-jumal* system, obtained from Al-Khwārizmī book, $9th$ century[26][27], Al-Kāshī book, $15th$ century [4], and Al-Yazdī book [28] in order to increase the diversity of our training examples. Hj-Tagged Corpus has 2,262 collected equations with 23,454 words and a vocabulary size of 5,049 words, have been manually tagged using our own tagset, the collected corpus is fully tagged with *O*

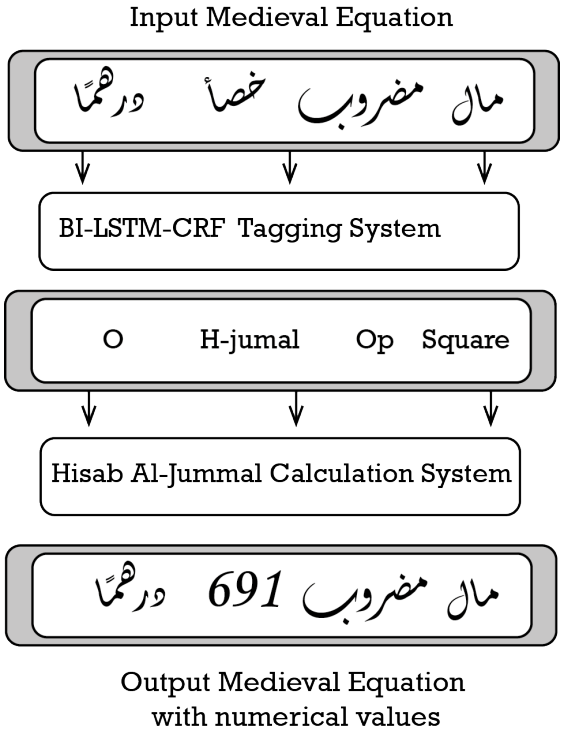

Figure 2: The proposed system architecture.

(*other*) or one of six entity tags: *H-jumal* (*hisāb al-jumal*), *Root* (*root*), *Square* (*square*), *Cube* (*cube*), *Equal* (*equal*), and *Op* (*operation*).

Table 3 shows some examples from the Hj-Tagged Corpus which consists of sequences of words and their tags. For evaluation, we report the precision, recall, and F1 scores for all tagged entities in the test set.

To ensure that the model does not see the context from the testing set during training, we first split the training, validation, and testing set on our collected dataset. The size of the split of our collected data into training, validation, and testing is 2,062, 100, 100 respectively.

4.2. Training

The Bi-LSTM-CRF model were implemented using the TensorFlow and Keras [29], a flexible neural network library written in Python. The general settings of our neural network model are listed below:

- Dimension of word embedding vector: 20.
- Dimension of hidden layer: 50 (for each LSTM: forward layer and backward layer).
- Learning method: SGD optimizer, learning rate: 0.01
- Number of examples used in each iteration(BATCH SIZE): 30.

Table 3
Some examples from the Hj-Tagged Corpus.

Medieval Arabic algebra Equation template	Tgas sequences (from left to right)
في ضرب فس في نفسه	O Op H-jumal O O
مال يعدل ضو أجذاره	Square Equal H-jumal Root
ؤ مال يعدل شش أجذار	H-jumal Square Equal H-jumal Root
مال يعدل عب	Square Equal H-jumal
مال و هغحرْ أجذاره يعدل إبل درهما	Square Op H-jumal Root Equal H-jumal O
قا مال و عض أجذاره يعدل بطزابا درهما	H-jumal Square Op H-jumal Root Equal H-jumal O
صقغ الكعب يعدل لصذ الأجذار	H-jumal Cube Equal H-jumal O
كعب و غد أجذاره يعدل مالان و هند من العدد	Cube Op H-jumal Root Equal Square Op H-jumal O O
مالان و انز مال المال تعدل ذكح كعب و خن أجذاره	Square Op H-jumal Square Square Equal H-jumal Cube Op H-jumal O
.....	.....

- We fix dropout rate at 0.5 for all dropout layers through all the experiments.
- Supervised learning was applied with up to 100 epochs for training the network.

4.3. Results and Discussions

In what follows (see Table 4), we use classification metrics such as precision, recall, and F1-score to evaluate our methods. F1-score can be computed to evaluate the performance of the system based on the detection results of the machine and the results of a human evaluator. F1-score is the harmonic mean of Precision and Recall computed from the number of mispronunciations detected by both the computer and human evaluator. They are defined as

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{1}$$

Precision = TP/(TP + FP) is the fraction of all positive predictions that are true positives, while Recall = TP/(TP + FN) is the fraction of all actual positives that are predicted positive. More precisely, the True Positive (TP), in this system, is the number of *Abjad* numbers the system got right, False Positive (FP) is the number of *Abjad* numbers wrongly selected, and False Negative (FN) are the *Abjad* numbers wrongly classified as no *Abjad* numbers.

First of all, we notice that Bi-LSTM-CRF network performs remarkably well on the Hj-Tagged Corpus with a mean F1 score of 98.1%. Additionally, using the same parameters, we compare Bi-LSTM-CRF model performance to a Bi-LSTM network. We show the precision, recall, and F1 scores of the models. One can see that adding CRF layer significantly improved prediction. Besides that, the training phase require less than 60 epochs to converge and it in general takes a few minutes. Finally, our experimental results suggests that Bi-LSTM-CRF network are less sensitive to training data size and the impact of noise from the tags.

Table 4
Precision, recall, and F1 scores of Bi-LSTM-CRF and Bi-LSTM models on the Hj-Tagged Corpus.

Model	Metric	
Bi-LSTM + CRF	Precision	99.0
	Recall	97.2
	F1	98.1
Bi-LSTM	Precision	97.1
	Recall	96.7
	F1	96.9

This paper focuses on recognizing and tagging components of a mathematical expression in medieval Arabic text. First, we want to mention here that our model was trained only on the Hj-Tagged Corpus. The training set is small, this limits the amount of ensemble diversity, which may reduce the network ability to generalize on new testing examples. Second, we did not perform any dataset preprocessing, apart from replacing every decimal number in the collected equations with a random numerical entity in the *hisāb al-jumal* system.

An other important point is that manually tagging such dataset with limited-vocabulary makes the system extremely sensitive to noise.

On the other hand, our model was able to correctly predict sentences which contain ambiguities in the test phase. For example, the word *wa* ("و") can mean the addition operator such as ("مال و يش أجذاره" [*Square Op H-jumal Root*]), or *hisāb al-jumal* entity such as ("مال و و أجذاره"[*Square Op H-jumal Root*]).

Finally, we implement a simple tag-based method to translate all *hisāb al-jumal* terms detected in the previous step to modern numeral system using the methodology described in Section 3. For example: The sentence of "يعدل س درهما مال و ج أجذاره" [*Square Op H-jumal Root Equal H-jumal O*], is transformed into: "مال و 11 أجذاره يعدل 60 درهما".

5. Conclusion

This paper experimented the first attempt to automatically analyze and recognize Abjad numerals in medieval Arabic mathematical texts using the Bi-LSTM CRF model. We also translate *hisāb al-jumal* terms detected in the previous step to modern numeral system. An additional key strength of this work is the time and effort spent on manually building a new dataset named Hj-Tagged Corpus, which consists of 2,262 tagged medieval mathematical sentences.

In the future, we can improve the intermediate representations learned in our network by training this model jointly with named entity recognition (NER) tags. We also plan to enrich the training examples by expanding Hj-Tagged Corpus. Another interesting direction is to apply our model to data from other Arabic sources in many different fields, such as geography, physics, chemistry, medicine, architecture, Astronomy, and so on.

Experimental results on the Hj-Tagged Corpus demonstrate that the proposed method offers

an important step in medieval Arabic mathematics analysis to enable scientists to understand and explore medieval mathematical texts.

References

[1] Chrisomalis, S. (2021) NUMERALS AS LETTERS: LUDIC LANGUAGE IN CHRONOGRAPHIC WRITING. 09.

[2] Farooqi, Mehr Afshan. (2003) "The Secret of Letters: Chronograms in Urdu Literary Culture." Edebiyat 13.2 : 147-58.

[3] Ifrah, Georges. (2000) "The universal history of numbers: From prehistory to the invention of the computer, translated by David Vellos, EF Harding, Sophie Wood and Ian Monk." .

[4] Miftāh al Hisāb Ghiyāth al-Dīn Jamshīd Mas'ud al-Kāshī, edited by: A.S al Damardāshī & all, Dar al Kitāb al 'Arabi, Le Caire, 1967, 357 pages

[5] Hinton, Geoffrey E., D. E. Rumelhart, and Ronald J. Williams. (1986) "Learning representations by back-propagating errors." Nature 323.9 : 533-536.

[6] Werbos, Paul J. (1988) "Generalization of backpropagation with application to a recurrent gas market model." Neural networks 1.4 : 339-356.

[7] AlKhwiter, Wasan, and Nora Al-Twairesh. (2021) "Part-of-speech tagging for Arabic tweets using CRF and Bi-LSTM." Computer Speech & Language 65 : 101138.

[8] Kamath, Shilpa, Chaitra Shivanagoudar, and K. G. Karibasappa. (2021) "Part of Speech Tagging Using Bi-LSTM-CRF and Performance Evaluation Based on Tagging Accuracy." Advances in Computing and Network Communications. Springer, Singapore. 299-310.

[9] Jin, Guozhe, and Zhezhou Yu. (2021) "A Korean named entity recognition method using bi-LSTM-CRF and masked self-attention." Computer Speech & Language 65 : 101134.

[10] Wintaka, Deni Cahya, Moch Arif Bijaksana, and Ibnu Asror. (2019) "Named-entity recognition on Indonesian tweets using bidirectional LSTM-CRF." Procedia Computer Science 157 : 221-228.

[11] Zeng, Donghuo, et al. (2017) "LSTM-CRF for drug-named entity recognition." Entropy 19.6 : 283.

[12] Wei, Qikang, et al. (2016) "Disease named entity recognition by combining conditional random fields and bidirectional recurrent neural networks." Database 2016 .

[13] Huang, Zhiheng, Wei Xu, and Kai Yu. (2015) "Bidirectional LSTM-CRF models for sequence tagging." arXiv preprint arXiv:1508.01991 .

[14] Anh, Le T., Mikhail Y. Arkhipov, and M. S. Burtsev. (2017) "Application of a Hybrid Bi-LSTM-CRF model to the task of Russian Named Entity Recognition." arXiv preprint arXiv:1709.09686 .

[15] Djamel, Hadj Mohammed, and Nacéra Bensaou. (2018) "Automatic Extraction of Equations in Medieval Arabic Algebra." 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA). IEEE.

[16] Gacek, Adam. Arabic manuscripts: a vademecum for readers. Vol. 98. Brill, 2009.

[17] Mikolov, Tomáš, et al. (2010) "Recurrent neural network based language model." Eleventh annual conference of the international speech communication association. .

[18] Mikolov, Tomáš, et al. (2011) "Strategies for training large scale neural network language models." 2011 IEEE Workshop on Automatic Speech Recognition & Understanding. IEEE.
[19] Jackson, Richard G., et al. (2017) "Natural language processing to extract symptoms of severe mental illness from clinical text: the Clinical Record Interactive Search Comprehensive Data Extraction (CRIS-CODE) project." BMJ open 7.1 : e012012.
[20] Swartz, Jordan, et al. (2017) "Creation of a simple natural language processing tool to support an imaging utilization quality dashboard." International journal of medical informatics 101 : 93-99.
[21] Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. (1994) "Learning long-term dependencies with gradient descent is difficult." IEEE transactions on neural networks 5.2 : 157-166.
[22] Hochreiter, Sepp, and Jürgen Schmidhuber. (1997) "Long short-term memory." Neural computation 9.8 : 1735-1780.
[23] Graves, Alex, and Jürgen Schmidhuber. (2005) "Framewise phoneme classification with bidirectional LSTM networks." Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.. Vol. 4. IEEE, .
[24] John Lafferty, Andrew McCallum, and Fernando CN Pereira. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of ICML2001, volume 951, pages 282–289.
[25] Lample, Guillaume, et al. "Neural architectures for named entity recognition." arXiv preprint arXiv:1603.01360 (2016).
[26] MUŠARRAFA, Alı Mustafı et AHMAD, Muhammad Mursı. Al-Khwārizmī. Kitab al-mukhtasar fı hisāb al-jabr wa'l-muqabalah, 1939.
[27] Roshdi Rashed, Al-Khwārizmī, Le commencement de l'algèbre, éd. 2009.
[28] "Muhammad Bāqir Zayn al-'Ābidīn al-Yazdī", 'Uyun al-Hissab (Les Fontaines du calcul), Manuscript undated - Harvard University, http://pds.lib.harvard.edu/pds/view/11328976?n=1&imagesize=1200&jp2Res=.25&printThumbnails=no.
[29] TensorFlow's implementation of the Keras, https://www.tensorflow.org/guide/keras,