Historical Event Detection and Extraction from Text

Grzegorz Meller, University of Milan

Introduction, Project Objectives and Literature Overview

Automatic historical information extraction from text is an interesting application of information retrieval and natural language processing. It can be used in various applications, for example for museum collections, allowing users to search for exhibits related to some historical events or actors within time periods and geographic areas.

The main goal of this project is to build a classifier that will be able to predict if text contains a historical event or not. In literature there are different definitions of historical event. The research paper "Historical Event Extraction from Text"[1] defines historical event as a model which consists of 4 slots: a location slot, time, participant and action slot. It is also important to distinguish texts, written shortly after an event happened, and the one that truly describes important, notable historical event. Therefore, it is important to filter out events that happened in the past, but it does not bring any historical value. Another goal of this project is to be able to retrieve entities that build up a historical information directly from the text by applying the named-entity recognition (NER). NER is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations, locations, or time expressions[2]. In the research paper: "Novel Event Detection and Classification for Historical Texts" there are presented several experiments that use NER technology in order to detect historical information in text by applying NER for sequence tagging. On the other hand, there are also many publications that seek for detecting historical events by building own knowledge bases and ontologies, like in the KYOTO project[3].

In this project, I perform several different experiments. First, I focus on building NER classifier, basing on the HISTO dataset. Then I build my own dataset, retrieving thousands of pages from Wikipedia using ontologies provided by DBpedia and build from scratch simple LSTM classifier, that will able to decide if text contains historical event (that truly have historical value) or not.

[1] Historical Event Extraction from Text. Agata Cybulska, Piek Vossen, VU University Amsterdam
[2] Named-entity recognition Wikipedia definition: https://en.wikipedia.org/wiki/Named-entity_recognition
[3] kyoto-project.eu

HISTO Dataset

The Histo corpus is dataset publicly available on GitHub[4] and it consist of historical texts of travel narratives and news published between the second half of the nineteenth century and the beginning of the twentieth century. This dataset is annotated according to specific guidelines precisely described in their repository, highlighting 22 distinct annotation classes (e.g. time, space movement, communication), where their existence testifies of the presence of a historical event information in a sentence. The dataset is very well preprocessed and organized, split into separate files for training and testing and represented in the BIO notation saved as .txt file. A fragment of the train.txt file is presented in the Fig.1.

```
Philosopher     O
,               O
reached         B-SPACEMOVEMENT
me              O
at              O
Gibraltar       O
,               O
and             O
served          B-ACTION
to              O
give            B-POSSESSION
me              O
a               O
homelike        O
feeling         B-EMOTIONSEVALUATIONS
```

Fig.1. Sentence “Philosopher, reached me at Gibraltar, and served to give me a homelike feeling” with annotations indicating existence of the historical event related to space movement, action, possession, and emotions.

The BIO notation is very often used for named-entity recognition tasks and is a common input format for deep learning models like artificial neural networks. The B- prefix before a tag indicates that the tag is the beginning of an annotated segment, and an I- prefix before a tag informs that the tag is inside the segment. The B- tag is used only when a tag is followed by a tag of the same type without O tokens between them. An O tag indicates that a token belongs to no chunk (no historical event mentioned)[5].

[4] Histo repository: https://github.com/dhfbk/Histo

[5] https://en.wikipedia.org/wiki/Inside%E2%80%93outside%E2%80%93beginning_(tagging)

CRF Classifiers

For the first set of experiments in automatically annotating text I implemented CRF classifier. The CRF stands for conditional random fields and is often applied for structured predictions, taking context (considering also "neighbor" words classification samples) into account, which makes it more advantageous over other classifiers. In natural language processing CRFs are often used for name entity recognition tasks. Besides words and corresponding annotations, it is possible to train the CRF classifier by enhancing input with extra features. In my example the features I used: word identity, word suffix, word shape and word POS (part of speech) tag, also, some information from nearby words is used (context is set to ±1, taking one word on left, and one on the right from the input word). When it comes to implementation details, no data preprocessing was needed, besides the one to extract data from text files. CRF classifier was implemented using the sklearn-crfsuite library[6]. Two experiments were performed, first considering only the event mention tag, so the model does not predict specific event class, but only that the word contains the event mention or not. In the second experiment the model predicts all 22 specific event classes. The results are presented in the Table 1.

TASK	P	R	F1
Mention detection only	0.828	0.772	0.799
Detection + Classification	0.586	0.492	0.517

Table 1: Performance of the CRF classifier, in terms of precision (P), recall (R), and F1-score.

Considering simplicity of that model, mention detection only results can be considered as good. Nevertheless, the results from the second experiment already shows the need of implementing more sophisticated classifier, that will be able to achieve much better results.

LSTM Approach

The second approach is based on the use of an implementation of LSTM model developed using TensorFlow library by Guillaume Genthial[7] that are later modified and customized for the purposes of this research project.

[6] https://sklearn-crfsuite.readthedocs.io/en/latest/index.html
[7] https://github.com/guillaumegenthial/tf_ner

Recurrent neural networks (RNN) have been employed to produce promising results on a variety of tasks in natural language processing. RNN maintains a memory based on history information, which enables the model to predict the current output conditioned on long distance features. Long Short Term Memory networks are the same as RNNs, except that the hidden layer updates are replaced by purpose-built memory cells. As a result, they may be better at finding and exploiting long range dependencies in the data[8]. What is more, in sequence tagging we not only have access to past input features but also to the future ones, so we can create bidirectional LSTM network. In doing so, we can efficiently make use of past features and future features at specific time frame. Coming back to the CRF classifiers, it is worth considering implementing them as the output layer of bi-LSTM, since CRF provide good functionality of directly connecting output tags with one another. Comparison of different models is presented in Fig.2.

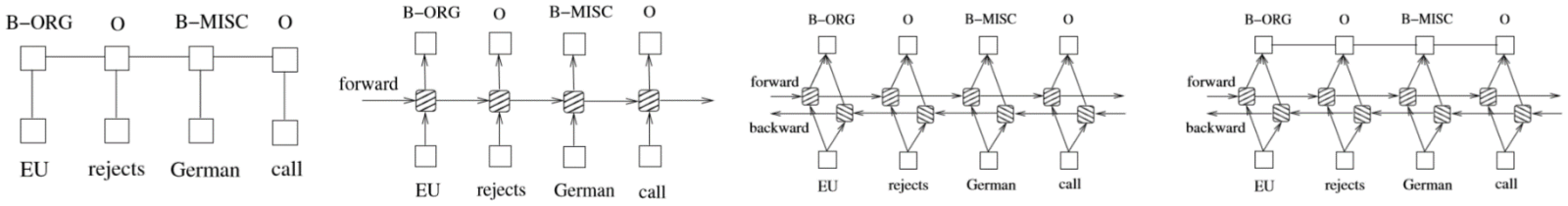

Figure 2: Simplified illustrations of different models for NER tasks, proposed in research paper "Bidirectional LSTM-CRF Models for Sequence Tagging". Going from left to right we start with CRF model, then LSTM network, bi-LSTM network and bi-LSTM-CRF network.

In order to start training the model, each word is mapped to a pre-trained word embedding (GloVe 840B) to represent every word in a digital form, as 50-dimensional vector. For training process, I choose the most complicated model, BI-LSTM-CRF. Results of predictions of that model are presented in the Table 2. As we can see results are much better comparing to the one when using only CRF classifier.

MODEL	P	R	F1
Mention detection only	0.942	0.945	0.943
Detection + Classification	0.891	0.901	0.891

Table 2: Performance of the bi-LSTM-CRF classifier, in terms of precision (P), recall (R), and F1-score.

[8] Bidirectional LSTM-CRF Models for Sequence Tagging, Zhiheng Huang, Wei Xu, Kai Yu

Considerations on the HISTO dataset and the trained model

Presented experiments on the HISTO dataset showed us great possibilities of building NER classifiers that can achieve relatively good results. Unfortunately, I must admit that the trained model on the HISTO dataset does not answer our main objective, which is deciding if an article describes notable historical event. Moreover, the model is not quite able to capture the most important historical information like place, date and people involved. HISTO dataset focus on the linguistic parts of speech that defines that some event happened in the past, but it does not relate to notable historical events. For example, the sentence taken from Wikipedia describing the historical event of battle of Marienburg:

"The Siege of Marienburg was an unsuccessful two-month siege of the castle in Marienburg (Malbork). The joint Polish and Lithuanian forces, under command of King Władysław II Jagiełło and Grand Duke Vytautas, besieged the castle between 26 July and 19 September 1410 in a bid of complete conquest of Prussia after the great victory in the Battle of Grunwald (Tannenberg)."

Uploading this sentence to the trained on the HISTO dataset model, gave the predictions of B-EXISTENCECAUSATION on the word *was* and B-HOSTILITY on word *Battle*. They are of course correct from the perspective of a language. *Was* indicates causations and *battle* indicates hostility but I would expect also other entities being retrieved that are more important from the perspective of the historical information. For example, I would like my model to retrieve words like *Marienburg, King Władysław II Jagiełło* and *26 July and 19 September 1410* as they build up the most important historical information.

In the next section we will focus on building another classifier, that will be trained on Wikipedia data, and will be able to answer if an article relates to a notable historical event, not just any event.

Building own dataset from Wikipedia

In this section I describe how to retrieve data from Wikipedia to build customized dataset that can be used for training the model. The goal is to retrieve about 11500 pages describing notable historical events, and about 11500 pages that do not have this information. Of course doing this manually would take too much time, fortunately there are knowledge bases that store ontologies that describe Wikipedia pages, thus it is possible to write efficient queries and retrieve pages relating to specific topics like history, biology, culture etc.

DBpedia is a project aiming to extract structured content from the information created in the Wikipedia project. This structured information is made available on the World Wide Web[9]. Thanks to that, it is possible to write SPARQL (like SQL) queries to DBpedia, retrieving Wikipedia pages that match defined metadata in a query.

[9] https://en.wikipedia.org/wiki/DBpedia

Using the rdflib python library (that allows to retrieve information from DBpedia directly from python console), Wikipedia pages were retrieved. Queries were constructed in a way that if page's rdf:type is equal to umbel-rc:ConflictEvent or yago:Siege or yago:War ontologies, than we are sure that important historical event is described in this page. Similarly, pages that do not describe historical events were retrieved, by searching for pages that have umbel-rc:Animal or dbo:Plant or yago:Wikicat20thCennturyFoxFilms or vago:Computer ontologies assigned. These ontology classes were not taken randomly. Animals and Plants are of course part of science, so model must learn how to distinct these pages from historical. 20th Century Films were selected, because model must learn that presence of dates does not always mean that historical event is described, because production of film or development of new type of computer is definitely event that happened in the past, but usually it is not something that contains historical value especially if it was developed/produced recently.

Finally, script saves to the csv file names of the pages (DBpedia link), abstracts and targets (1 if page describes historical event, 0 otherwise). From text of a Wikipedia page only abstract is saved, since it contains the most important information, and based on that, model should be able to make a good prediction. What's more training on less text, significantly improve the learning speed.

LSTM network for detecting pages describing historical events

First, data needs to be preprocessed. Punctuation, URLs and stop words are removed. Then tokenization is applied, and dataset is split into train and test set (with 0.8 ratio). Then all text sequences are padded, so they all have the same size (150 words which is approximate length of most Wikipedia abstracts).

Model starts with an embedding layer. The embedding layer stores one vector per word. When called, it converts the sequences of word indices to sequences of vectors. The next layer is the LSTM layer that propagates the input forward and concatenates the output. This helps the RNN to learn long range dependencies[10]. Then the final output layer with sigmoid activation function outputs the result. Because we deal here with 0/1 classification, selected loss function is binary cross entropy. Optimizer is Adam, which is the type of stochastic gradient descent. Results are presented in the Table 3.

MODEL	P	R	F1
LSTM binary classifier	0.989	0.969	0.979

Table 3. Model evaluation on the test set.

[10] Based on https://www.tensorflow.org/tutorials/text/text_classification_rnn

Results look very promising, but we must remember, that the model was trained on about 22000 pages that describe only a small portion of all possible Wikipedia sites. Therefore, second test set was created, but on pages that are assigned to different ontologies, than those on which the model was trained. This allows to find out if the model generalizes well on different data. New test set contains more than 1170 pages, were 587 pages describe historical events of Olympic games, Nazi crimes in the WW2, revolutions, uprisings, and the newest, yet still important and notable historical events of the European Union. Another 586 pages describe artificial neural networks, rain forests, food, and sports activities descriptions. As we can see new test set contains different page types than one on which the model was trained. The results of classification on the new test set are presented in the Table 4. Results are lower, but we must remember that this test was performed on completely different dataset. Results are still acceptable, considering simplicity of the model, therefore we can say that this approach is able to generalize. Possibly creating training dataset more precisely, with very in-depth selection of ontologies could even increase overall results.

MODEL	P	R	F1
LSTM binary classifier	0.912	0.771	0.836

Table 4. Model evaluation on different data, extracted from different set of ontologies then those on which the model was trained.

Applying NER to pages classified as historical using SpaCy library

In this last short experiment, I would like to show that to locate in text common entities, like time or locations, it is not always required to build and train a deep learning model from scratch. Nowadays, there are many NLP libraries that can perform NER tasks well. Figure 3 presents experiment on already presented text of the battle of Marienburg.

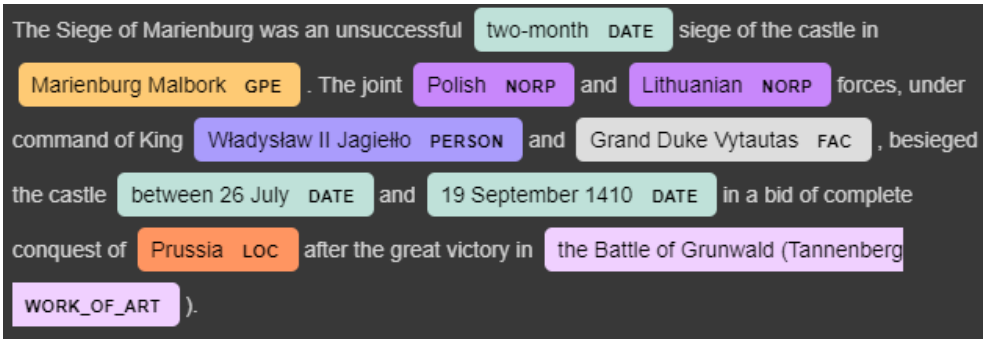

Fig. 3. SpaCy NER example

SpaCy NER was able to correctly display dates (DATE), nationalities (NORP) and places (tagged as GPE). It was not able to correctly highlight the event. Even though this tagging is not 100% correct it adds a good value by extracting the most important historical information.

Final Conclusions

In this project I presented several approaches towards building a model for historical event detection and historical information extraction. First model, trained on the HISTO dataset, is able to tag in a text linguistic parts of speech that indicate the existence of a past event. Second model, built on self-created Wikipedia dataset, can decide if a given text consist of historical value or not. This experiment also shows how great the usage of DBpedia ontologies can be for creating custom datasets for training deep learning models. Two models combined could be a powerful tool, that will first detect if model describe important historical event and then tag its linguistic parts of speech to highlight for example the hostility of two participants of the event. Unfortunately, the HISTO dataset does not highlight the most important historical entities, which are location, time, participants, and action (as defined in the introduction). But the short case study of SpaCy shows that extraction of these entities is possible to a certain extent with popular NLP libraries. If I were to continue this project, the next steps would be to create dataset based on Wikipedia in the BIO format (so it would be similar to the HISTO dataset) and tag most important entities that build historical event. This task probably could not be fully automatized, and would require some manual tagging, because DBpedia ontology does not provide appropriate ontology class to every word with the indication of its location. The creation of a good dataset answering the needs of the project and training a model like bi-LSTM-CRF based on that data, could lead to a great final results of a model that would be able to extract the most important historical information from text.

References

1. Historical Event Extraction from Text. Agata Cybulska, Piek Vossen. VU University Amsterdam.
2. Novel Event Detection and Classification for Historical Texts. Rachele Sprugnoli, Sara Tonelli.
3. Bidirectional LSTM-CRF Models for Sequence Tagging. Zhiheng Huang, Wei Xu, Kai Yu.
4. Sequence Tagging with TensorFlow. Guillaume Genthail.