TÉCNICO LISBOA

Toponym Resolution in Text with Neural Language Models

Diogo Alexandre Araújo Viegas

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Bruno Emanuel da Graça Martins
Dr. Maguelonne Teisseire

Examination Committee

Chairperson: Prof. Mário Jorge Costa Gaspar da Silva
Supervisor: Prof. Bruno Emanuel da Graça Martins
Member of the Committee: Dr. Jacques Fize

November 2021

Acknowledgments

First and foremost, I would like to thank my advisor Professor Bruno Martins and also Professor Jacinto Estima, for providing me guidance throughout this work and devoting an effort to the success of this thesis. Their knowledge and expertise in the field of geographic information systems is incredibly inspiring and motivates me to always pursue my best work. Additionally, I would like to thank Doctor Maguelonne Teisseire and Doctor Mathieu Roche for co-supervising my project. A special thanks to Doctors Ludovic Moncla and Jacques Fize for providing comments on both the project thesis discussion and the final thesis presentation, respectively.

I would like to thank my family, specially my parents and sister, for their continuous support and understanding during my research and writing my project. A special thanks to my girlfriend Mariana for being the most amazing and supportive person I could ask for.

I am also thankful for having met incredible people throughout my journey at Instituto Superior Técnico. We went through it all over the past five years, but I consider myself a privileged to have private with them. Their motivation and encouragement was essential during this time.

Abstract

Toponym resolution addresses the task of mapping names for places (toponyms), previously detected in a textual document, into the corresponding locations on Earth (i.e., through geographical coordinates). This process is important in multiple applications in the context of digital humanities, computational social sciences, and other scientific domains (e.g., epidemiology, etc.). Complexity arises when a single place name refer to many real locations (e.g., *Paris* can refer to more than 50 places in over 20 countries around the world), making disambiguation a non-trivial procedure. I present a novel toponym resolution method based on deep learning, taking inspiration from recent methods achieving state-of-the-art results. The proposed neural network architecture is based on a pre-trained language model, sharing parameters for the processing of different inputs (e.g., the toponym to disambiguate along with the surrounding words). The HEALPix method is used to model toponym resolution as a classification task, by sub-dividing the Earth's surface into a set of cells. Subsequently, the result of the classification is used to inform the prediction of geographic coordinates for each place name reference, through a separate layer that directly applies the great circle distance as a loss function. Additionally, I also test the use of external geophysical information, by using an additional term in the cost function for each geophysical property considered, thus allowing the model to obtain more information about the locations when making predictions. The proposed model was tested on collections of documents used and developed in previous studies. The obtained results show that the proposed model can significantly outperform previous approaches.

Keywords

Geographical Text Analysis; Toponym Resolution; Deep Learning; Natural Language Processing; Neural Language Models; Geophysical Properties

Resumo

A resolução de topónimos aborda a tarefa de mapear nomes de lugares (topónimos), previamente detetados num documento textual, nos locais na Terra correspondentes (i.e., através de coordenadas geográficas). Este processo é importante em múltiplas aplicações no contexto das humanidades digitais, ciênciais sociais computacionais, e outros campos cientificos (e.g., epidemiologia, etc.). A complexidade surge quando um único nome de lugar se refere a muitos lugares reais (e.g., Paris pode-se referir a mais de 50 lugares em mais de 20 países em todo o mundo), tornando a sua desambiguação um procedimento não trivial. Eu apresento um novo método de resolução de topónimos baseado em aprendizagem profunda, tendo como inspiração os mais recentes métodos desenvolvidos alcançando resultados estado-da-arte. A arquitetura da rede neuronal proposta foi baseada num modelo de linguagem pré-treinado, partilhando parâmetros para o processamento de diferentes entradas de texto (e.g., o input a desambiguar, bem como as palavras circundantes). O método HEALPix é usado para modelar a tarefa de resolução de topónimos como uma tarefa de classificação ao sub-dividir a superfície da Terra em conjuntos de células. Subsequentemente, o resultado da classificação é usado para informar a previsão de coordenadas geográficas para cada referência de nome de lugar, através de uma camada que aplica diretamente a distância do grande círculo como uma função de perda. Adicionalmente, eu também testo o uso de informação geofísica externa, ao adicionar um termo à função de custo correspondente a cada uma das propriedades geofísicas consideradas, permitindo ao modelo obter mais informação sobre as localizações aquando as previsões. O modelo proposto foi testado em coleções de documentos usados e desenvolvidos em estudos anteriores. Os resultados obtidos demonstram que o modelo proposto pode gerar resultados substancialmente melhores que abordagens anteriores.

Palavras Chave

Análise Geográfica de Texto; Resolução de Topónimos; Aprendizagem Profunda; Processamento de Língua Natural; Modelos de Linguagem Neuronal; Propriedades Geofísicas

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

BERT	Bidirectional Encoder Representations from Transformers
BPTT	Backpropagation Through Time
CBOW	Continuous Bag of Words
CNN	Convolutional Neural Network
ED	Entity Disambiguation
EL	Entity Linking
ER	Entity Recognition
ELMO	Embeddings from Language Models
HEALPix	Hierarchical Equal Area isoLatitude Pixelization
KB	Knowledge Based
LGL	Local Global Lexicon
LSTM	Long Short-Term Memory
MLG	Multi-Level Geocoder
MLM	Masked Language Modeling
MLP	Multilayer Perceptron
NLP	Natural Language Processing
NSP	Next Sentence Prediction
RNN	Recurrent Neural Network
vMF	von Mises-Fisher
WOTR	War of the Rebellion

1

Introduction

Contents

Toponym resolution, also known as geo-parsing, geo-grounding or place name resolution, aims to assign unambiguous locations (e.g., geographic coordinates) to all location names mentioned in documents. Toponym resolution is usually performed in two independent steps. The first step concerns toponym detection, where the spans of text corresponding to place names mentioned in a document are delimited. In the second step, toponym disambiguation or geocoding, each of the discovered place names can be mapped to latitude and longitude coordinates corresponding to the centroid of its physical location. A related problem to toponym resolution is document geocoding, which concerns with assigning an entire textual document to the latitude and longitude coordinates that best match its contents, e.g. summarizing the results of toponym resolution or instead directly leveraging the textual contents.

The names of locations are usually ambiguous expressions, since these names can correspond to distinct geographical referents (e.g., the place name *Paris* is associated to several geographic locations, besides the capital city of *France*), and the same location can often be associated to several alternative names (e.g., the names *New York* and *Big Apple* can both be used as references to *New York City* in the *United States*).

Toponym resolution is important in many real-life applications in the field of digital humanities, computational social science [50], and other scientific domains, namely epidemiology, which has been an hot topic recently. One aim in epidemiology is to create maps for the locations of specific health problems, which provide tools to monitor and intervene during disease pandemics. Other applications can also benefit from the improvement of the results in the toponym resolution task, such as the improvement of search engine results (e.g., by geographic indexing or search results clustering), or document organization according to spatial criteria. Additionally, Internet-enabled mobile devices are increasingly common which increases the requirement for location-based services and other highly local content (i.e., information that is relevant to where users are, or the places in which they are interested) [22].

Many previous methods for toponym resolution have explored the use of heuristics, relying on an external source of knowledge (e.g., a gazetteer) to assign textual references to corresponding locations [19, 20]. More recently, several studies have considered supervised learning approaches that take heuristics as features in machine learning models [22, 39, 47], while later studies have explored the use of the text of a document within the surrounding context, through statistical language models [4, 50].

Nowadays, the task of toponym resolution can also be addressed through the use of state-of-the-art deep learning methods for Natural Language Processing (NLP), e.g. related to the use of contextual embedding models such as Bidirectional Encoder Representations from Transformers (BERT) or RoBERTa [25]. These neural methods offer several advantages over existing rule-based techniques for toponym resolution, namely the ability to naturally leverage contextual clues to improve predictions and disambiguate location names. However, previous studies have shown that the performance of these methods varies greatly when applied to corpora of different genres and domains.

1.1 Objectives

This M.Sc. thesis proposes a novel method for toponym resolution, using a Transformer-based pre-trained language model (namely the BERT and RoBERTa models), that builds a representation of the toponym to be disambiguated together with its surrounding context. Other toponyms, or even regular words appearing in the surrounding context, can be characteristic of a certain region, which can provide clues about the location of the mention (e.g., the words *Louvre* or *Seine* are usually associated with the toponym *Paris*, and the words *fashion* or *football* are usually associated with the toponym *Milan*).

The goal is to analyze how this method performs when evaluated on previous works datasets (i.e., scientific publications and general news texts) [5, 10, 23, 29, 48], used and developed in previous studies, comparing the results with state-of-the-art approaches [3, 5, 10, 18, 39].

1.2 Methodology

The first stage of my work is to divide the original documents into the different textual inputs, namely the target toponym to be disambiguated, the close context surrounding the target toponym, and the wider-range context surrounding the target toponym. Each of these textual inputs is then fed into a pre-trained language model, sharing parameters for the processing of the three sequences.

After that, I use the Hierarchical Equal Area isoLatitude Pixelization (HEALPix) [9] scheme to partition the geographic study region into a set of cells, in order to approach the task of toponym resolution as a classfication task, by associating each place name reference with a region on the surface of the Earth. Furthermore, a regression loss based on the great circle distance (i.e., the shortest distance between two points on the surface of a sphere) is used for the output corresponding to the geographical coordinates of the place name reference.

Additionally, I test the use of external information corresponding to geophysical properties (e.g., terrain properties, natural resources, etc.) in order to turn the system more robust and improve performance. This information is extracted from external raster datasets and incorporated in the proposed model, through an additional term in the cost function for each geophysical property considered. My goal with this experiment is to enable the model to obtain more information about the locations when making predictions.

The proposed model achieves interesting results in most of the experiments. In some cases, state-of-the-art results are obtained. Thus, one can argue that fine-tuning pre-trained language models, as performed in this dissertation, can be effective on the task of toponym resolution.

1.3 Organization of the Document

The rest of this document is organized as follows. In Section 2, fundamental concepts necessary to better understand the rest of the document are introduced. Section 3 describes some of the previously developed related studies in the field of toponym resolution. Section 4 details the proposed model as well as the performed experiments. Meanwhile, section 5 presents the experimental evaluation, in addition to the obtained results. Finally, section 6 summarizes the conclusions and presents ideas for future work.

2

Fundamental Concepts

Contents

This chapter discusses fundamental concepts for a clearer understanding of the work developed in this dissertation. Section 2.1 explains fundamental concepts regarding neural networks and deep learning, together with strategies regarding the training of these models. In particular, deep neural networks architectures that are widely used in text classification are debated, such as convolutional and recurrent neural networks as well as the Transformer model. Section 2.2 describes the representation of text for automatic classification, from approaches based on one-to-hot representations to the most novel BERT pre-trained language model, passing through the Word2Vec algorithm. Finally, Section 2.3 provides an overview of the topics that were discussed and presented in this chapter.

2.1 Deep Learning for Natural Language Processing

Neural networks are inspired by biological neural networks and are a class of algorithms whose main goal is to perform learning. This is done by processing examples or instances, each containing a known input and output, and forming probability-weighted associations between the two, which are stored within the data structure of the network itself. The training of a neural network is usually performed by determining the difference between the processed output of the network (often a prediction) and a target output (the known result), afterwards adjusting the weighted associations accordingly. Successive adjustments will cause the neural network to produce an output which is increasingly similar to the target output. After a sufficient number of these adjustments the training can be terminated based upon certain criteria. Deep learning is known as the process of learning using a neural network with multiple layers of processing units.

2.1.1 Multilayer Perceptron

It is said that the Perceptron model is the predecessor to neural network algorithms, as it was first introduced by Frank Rosenblatt in 1958. This is a binary classifier that receives a set of input values and outputs 0 or 1. The Perceptron introduces the concept of weights (i.e., a numerical representation of the importance of the corresponding input) in order to produce an output which is the weighted sum of the inputs. Assuming the input as x_n and its corresponding weight as w_n as well as b as a bias term, the Perceptron can be represented through the following equation:

$$f(\boldsymbol{x}) = \hat{y} = \sigma\left(\sum_{n=1}^{M} x_n \cdot w_n + b\right) \tag{2.1}$$

In the previous equation, M is the number of input features, and σ represents the activation function, this latter concept is also fundamental in the context of neural networks, corresponding to arrangements of processing units similar to the Perceptron. The activation function's purpose is to introduce non-

linearity into the model, which means that the output cannot be reproduced from a linear combination of the inputs, and make the model capable to learn and perform more complex tasks.

There are a considerable number of activation functions that map a given value into a certain range. Some of the most commonly used activation functions are:

- Sigmoid - The sigmoid activation function maps each value x into a $[0, 1]$ range. The mathematically expression for the sigmoid function is:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

- Hyperbolic tangent - The hyperbolic tangent activation function maps each value x into a $[-1, 1]$ range. The mathematically expression for the hyperbolic tangent function is:

$$\tanh(x) = 2 * \sigma(2x) - 1 \tag{2.3}$$

- Rectifier - The rectifier activation function returns 0 if it receives any negative input, but for any positive value x it returns that value back. The mathematically expression for the rectifier function is:

$$ReLU(x) = \max(0, x) \tag{2.4}$$

As said before, after determining the difference between the generated output and the target one, we can adjust the weighted connections. This is known, in the simpler case, as the Perceptron learning rule, and it can be written as:

$$\boldsymbol{w_{n+1}} = \boldsymbol{w_n} + \gamma \cdot (y - \hat{y}_n) \cdot \boldsymbol{x} \tag{2.5}$$

In the previous equation, n and $n + 1$ represent consecutive versions of the model weigths, and γ represents a learning rate, i.e. an hyperparameter that controls how much to change the model in response to the estimated error, each time the model weights are updated.

The Perceptron can be seen as a single neuron model. A neural network can be, and usually is, composed of more than one neuron. These neurons can be grouped into layers with connections between them, in the sense that the input of a neuron in a layer can be the output of a neuron in a previous layer. The architecture based on this concept of multiple layers with multiple neurons each, is known as a Multilayer Perceptron (MLP). MLP's are also known as feedforward neural networks, in that each layer is fed with information that flows from the input layer to the output layer, passing through the

hidden layers. The learning in this model occurs by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. This process is done in a backpropagation phase, by computing the gradient of the loss function concerning the weights of the network, to achieve better results than the previous version, by applying the gradient descent algorithm. The following formula is applied:

$$\boldsymbol{\Delta w_{n+1}} = \boldsymbol{\Delta w_n} - \gamma \cdot \boldsymbol{\nabla_w L\left(\Delta w_n\right)} \tag{2.6}$$

In the previous equation, ∇_w represent the gradient of the loss function $L()$ with respect to the weight parameters (Δw_n).

There are many variations of gradient descent. The most straightforward is the batch gradient descent, which uses the whole dataset on each iteration of the parameter updates. It has a straight trajectory towards the minimum and, in theory, is guaranteed to converge to the global minimum, if the loss function is convex, and to a local minimum, if the loss function is not convex. However, batch gradient descent may be slow to go over all examples, especially when we have large datasets. Having in mind that some samples do not contribute that much to the parameters update, it may also constitute a waste of resources.

Another variation is the mini-batch gradient descent. Instead of going through all the examples, like the batch gradient descent, it sums up over a fixed number of examples, i.e., the size of the batch. By learning only over a fixed number of examples, it makes the process faster when compared to the previous approach. Another advantage is that it adds noise to the process, which may help improving the generalization error, avoiding overfitting. Nevertheless, by having fewer examples, the error is estimated to be higher than in the previous approach, since there are fewer guarantees regarding it converging to the minimum. This particular proposal offers a neutral ground between efficiency and effectiveness.

Stochastic gradient descent is another alternative that consists of doing the update on each entry of the dataset. This yields the highest efficiency among all the approaches, but it also increases variance since we only use one example for each learning step.

2.1.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a type of deep neural network designed to receive structured data as input, usually images [32]. However, it was discovered that this type of architecture has been achieving state-of-the-art results when dealing with sequential data, hence becoming a force to be reckoned with in NLP. It is important to note that in NLP, instead of image pixels, the input usually consist of documents or sentences, represented as matrices. The vectors represented in the matrix are typically word embeddings (which will be discussed later on).

CNN's are a way to structure a network's architecture such that it uses structured knowledge from how inputs are built to reduce the number of free parameters and thus increase generalization. The foundation of these networks is the convolution operation, usually represented by the symbol $*$. In the case of two-dimensional inputs, given a $k_h \cdot k_w$ kernel matrix K_{ij} and a input matrix $X \in \mathbb{R}^{HxW}$, the convolution of X by K is a matrix $O = X * K$ where the coordinates are:

$$O_{ij} = (X * K) = \sum_{h=1}^{k_h} \sum_{w=1}^{k_w} X_{(i+h-1)(j+w-1)} K_{hw} \tag{2.7}$$

Convolutional layers are often used together with a non-linear activation function, providing some non-linearity to the network. Most often, the ReLU function is used keeping the positive values.

Another common operation within CNN's is the pooling operation, which is very similar to the previously explained convolution operation although not involving a weight kernel. For each window in the input, a fixed operation is applied. Usually, two kinds of pooling are used: max pooling and average pooling. This process is normally done to change the resolution of the representation, summarizing the main information. This not only allows for noise to fade but also yields a smaller representation, which will result in fewer parameters and thus more generalization. However, if the pooling operation is overdone, the resolution is too small and we may lose not only noise but also very important features.

In a CNN, the final computed stack of feature maps is usually flattened and fed as input to one or more fully-connected layers, that return the final classification. A loss over this classification output is then used to train the network with the backpropagation algorithm.

2.1.3 Recurrent Neural Networks

Recurrent Neural Network (RNN)'s are a family of neural networks designed to process sequential data with variable length, being thus extremely suitable for NLP tasks [32], given the need for processing sequences of words. A traditional fully connected feedforward network would have separate parameters for each input feature. By comparison, an RNN shares the same weights across several time steps. Each member of the output is a function of the previous members and is produced using the same update rule applied to the previous outputs.

More formally, RNN's receive as input a sequence of variable length vectors $\boldsymbol{x_i}, ..., \boldsymbol{x_n}$, and return as output a single vector $\boldsymbol{y_n}$ (e.g., for classification tasks) or a sequence of vectors. An RNN can be defined recursively, according to a function $\mathrm{R}()$ that takes as input a state vector $\boldsymbol{h_{i-1}}$ and an input vector $\boldsymbol{x_i}$, and returns a new state vector $\boldsymbol{h_i}$. The state vector is then mapped to an output vector using a simple deterministic function $\mathrm{O}()$ [8]. Figure 2.1 shows a representation of a vanilla RNN architecture, where we can see recursion in the parameters used at each step of the model. This recursion can be unrolled

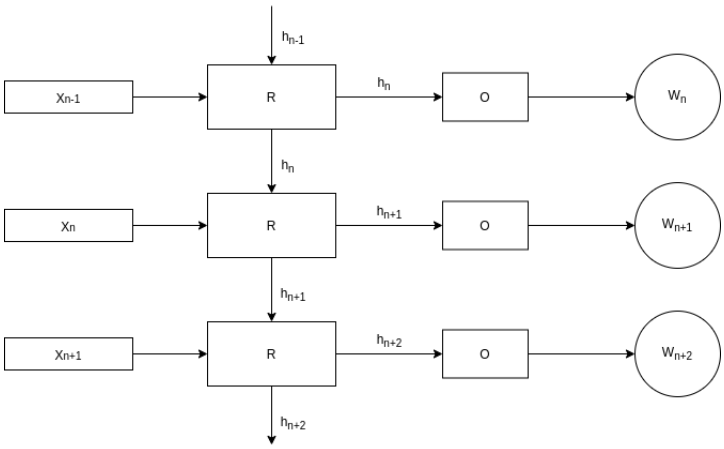


Figure 2.1: Vanilla RNN architecture.

such as:

$$\begin{aligned} \boldsymbol{h_{n+2}} &= \mathrm{R}(\boldsymbol{h_{n+1}}, \boldsymbol{x_{n+1}}) \\ &= \mathrm{R}(\mathrm{R}(\boldsymbol{h_n}, \boldsymbol{x_n}), \boldsymbol{x_{n+1}}) \\ &= \mathrm{R}(\mathrm{R}(\mathrm{R}(\boldsymbol{h_{n-1}}, \boldsymbol{x_{n-1}}), \boldsymbol{x_n}), \boldsymbol{x_{n+1}}) \end{aligned} \tag{2.8}$$

It is also easy to observe that an unrolled RNN is just a very deep neural network. To train this network, it is necessary to add a loss node to the unrolled graph, and then use the backpropagation algorithm to compute the gradients concerning that loss. This is called the Backpropagation Through Time (BPTT) method. The algorithm is the same as the standard backpropagation algorithm that we explored in the case of deep feedforward neural networks, with a key difference in that its sums up the gradients for weights W at each time step. A traditional neural network does not share parameters across layers, so there is no need to sum. When applying BPTT, it is common to have very large sequences, and thus going backward through an entire sequence on each backward pass can be computationally expensive. Also, this causes problems when training over a long sequence, since computing the gradient of the cost for one of the first hidden states requires successive multiplications by the same weight matrix. This can cause the gradients to either explode (i.e., if the largest singular value of the matrix is larger than one) or vanish (i.e., if the largest singular value of the matrix is less than one), preventing the network weights from adjusting as needed. To avoid this sort of issue, different RNN architectures have been developed.

One of these architectures is the Long Short-Term Memory (LSTM) architecture, which was designed to better capture long term dependencies and address the gradient vanishing problem, by introducing a memory cell to remember values over arbitrary time intervals, together with three gates (input, output, and forget gate) to regulate the flow of information into and out of the memory cell [32]. Having that in

mind, LSTM cells can be defined as follows [45]:

$$
\begin{aligned}
\boldsymbol{i} &= \sigma(W_{hi} \cdot \boldsymbol{h_{t-1}} + W_{xi} \cdot \boldsymbol{x_t} + \boldsymbol{b_i}) \\
\boldsymbol{f} &= \sigma(W_{hf} \cdot \boldsymbol{h_{t-1}} + W_{xf} \cdot \boldsymbol{x_t} + \boldsymbol{b_f}) \\
\boldsymbol{o} &= \sigma(W_{ho} \cdot \boldsymbol{h_{t-1}} + W_{xo} \cdot \boldsymbol{x_t} + \boldsymbol{b_o}) \\
\boldsymbol{g} &= \tanh(W_{hg} \cdot \boldsymbol{h_{t-1}} + W_{xg} \cdot \boldsymbol{x_t} + \boldsymbol{b_g}) \\
\boldsymbol{c_t} &= \boldsymbol{f} \odot \boldsymbol{c_{t-1}} + \boldsymbol{i} \odot \boldsymbol{g} \\
\boldsymbol{h_t} &= \boldsymbol{o} \odot \tanh(\boldsymbol{c_t})
\end{aligned}
\tag{2.9}
$$

In the previous equation, the gates $\boldsymbol{i}$, $\boldsymbol{f}$, and $\boldsymbol{o}$ use the sigmoid activation function, while the gate $\boldsymbol{g}$ uses the hyperbolic tangent activation function. The parameters, $\boldsymbol{b}$ and $\boldsymbol{c}$ represent bias vectors, while W represents the weight matrices.

2.1.4 Transformer Models

Even though LSTM models have improved performance regarding increased length sentences, the same problem that happens with RNN's usually happens with LSTM's. When sentences are too long, the model usually forgets the content of distant positions in the sequence. Another problem with RNN variants is the fact that it is almost impossible to parallelize sentence processing, since there is a need to process word by word.

Transformers overcome this liability by applying attention modules, more specifically multiple self-attention operations, to process in parallel every word in a sentence or document. Attention scores are used to model the influence each word has on other [32].

The use of attention in a neural network, is motivated by how we pay visual attention to different regions of an image, or correlate words in one sentence. Human visual attention allows us to focus on a given detail with *high resolution* (i.e., more attention) while perceiving the surrounding image in *low resolution* (i.e., less attention), and then adjust the focal point or perform the inference accordingly. In NLP, the attention mechanism was first introduced to help memorize long source sentences in neural machine translation. It can be interpreted as a vector of importance scores. To predict a word in a sentence, using the attention vector, we estimate how strongly this output is connected with other words, and take the sum of their values weighted by the attention vector as the approximation of the target.

Regarding the Transformer architecture, one might say that it follows the most competitive neural sequence transduction models by having an encoder-decoder structure. This structure consists of an encoder that maps an input sequence $\boldsymbol{x_i}, ..., \boldsymbol{x_n}$ into a sequence of continuous representations $\boldsymbol{z_i}, ..., \boldsymbol{z_n}$, and a decoder that generates an output of symbols $\boldsymbol{y_i}, ..., \boldsymbol{y_n}$ one at a time. In the original Transformer proposal, the encoder is composed of a stack of 6 identical layers, each containing 2 sub-layers: a

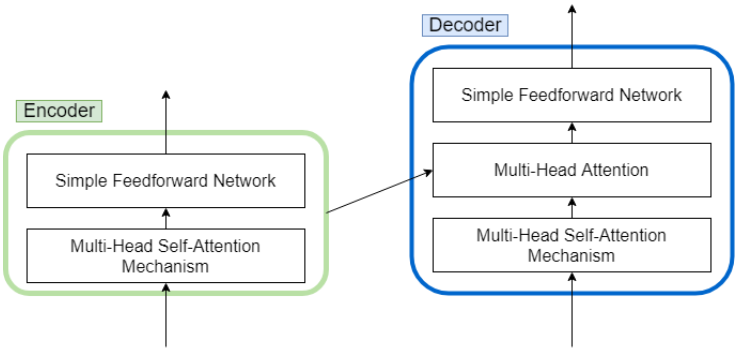


Figure 2.2: General structure of each Transformer encoder (left) and decoder (right).

multi-head self-attention mechanism and a simple feedforward network. On the other hand, the decoder is also composed of a stack of 6 identical layers, but each contains 3 sub-layers: the same 2 as the encoder, plus another multi-head attention over the output of the encoder stack [46]. Figure 2.2 shows the architecture of the Transformer model.

In detail, the first of the six encoders is the only one receiving embeddings as input (i.e this layer receives vector representations for the input tokens, while the following encoder layers receive the output of the previous one).

The first step in the self-attention layer is creating three vectors (query, key, and value vector) for each input. These vectors are created by multiplying the input vector by three matrices that are simultaneously trained. The second step is to calculate a self-attention score. This score is achieved by calculating the dot product between the query vector and the key vector. The next steps (third and fourth) consist of dividing the score by the square root of the dimensionality of the key vectors, and passing that result through the softmax function. The fifth step is to multiply the softmax function output by each value vector. Finally, each weighted vector is summed. This produces the output of the self-attention layer for each word.

The authors described the aforementioned process as scaled dot-product attention, and it can be expressed as shown in the following equation:

$$\text{Attention}(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}) = \text{softmax}\left(\frac{\boldsymbol{q}\boldsymbol{k}^T}{\sqrt{d_k}}\right)\boldsymbol{v} \tag{2.10}$$

There is also the concept of multi-head attention which expands the model's ability to focus on different positions. This gives the attention layer multiple *representation subspaces*, as we have multiple sets of query/key/value weight matrices, with one set for each attention head.

All the attention heads that were previously calculated through equation (2.10) are concatenated and multiplied by a matrix W^O that is trained jointly with the attention model. The result is a matrix Z that captures the information from all the attention heads and is then passed to the feedforward layer.

$$\begin{aligned} \text{MultiHeadAttention}(q, k, v) &= \text{concat}(Head_1, ..., Head_h)W^O \\ \text{where} \quad Head_i &= \text{Attention}(\boldsymbol{q}W_i^q, \boldsymbol{k}W_i^k, \boldsymbol{v}W_i^v) \end{aligned} \tag{2.11}$$

In addition to attention sub-layers, each of the layers in the encoder and the decoder contains a fully connected feedforward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation function in between.

The decoder stack outputs a vector of floats that need to be transformed into a word. For that to happen, a linear layer is used to transform the vector into a logits vector with the size of the vocabulary. After that, the vector passes through a final softmax operation to turn those scores into probabilities. The cell with the highest probability is chosen, and the word associated with it is produced as the output for the time step.

Finally, one key component that is still missing from the architecture is the notion of position within a sequence, given that is not the same to process tokens at the beginning, at the end, or at the middle of the text. To inject the notion of position into the embedding that represent the token, the authors added *positional encodings* when encoding each word. *Positional encodings* are vectors with the same size as the embeddings that, using the cyclic nature of the $\sin()$ and $\cos()$ functions, represent information about the position of a word in a sequence.

2.2 Representing Text for Automatic Classification

In text analysis tasks, it is necessary to represent text as matrices, since machine learning algorithms do not take plain text as input. In order to do this, each word needs to be assigned a numerical vector. There are several approaches, and some of them will be presented in the following subsections.

2.2.1 Traditional Text Representation Approaches

A traditional approach for representing text is the one-hot encoding, which models the presence of a word from a given vocabulary through binary design (i.e., 1 if the word occurs and 0 if it doesn't). Individual words can thus be represented through binary indicative vectors with a dimensionality equal to the size of the vocabulary, and documents can be represented through aggregations of the words representations, through addition or boolean operations.

Even though this is probably the simplest approach to the aforementioned problem, it has quite a few downsides. One issue is the curse of dimensionality (i.e., various problems that arise when analyzing and organizing data in high-dimensional spaces) since one-hot encoding generates vectors with a fixed size of $|V|$ (i.e., vocabulary size), meaning that a word is represented by a single 1 and $|V|$-1 0's if present

in the document, becoming a sparse representation. This means that a word in this representation has no meaning, and the notion of similarity between words does not exist. An additional problem is the fact that only words present in the vocabulary can be represented by this model, being impossible to represent different words in previously unseen text.

2.2.2 Word Embeddings

Another way of representing text is through word embedding. This method consists in learning to map words into real number vectors, and seizing linguistic information by capturing similarity between words. Word vectors are positioned in the vector space in a meaningful way so that distance between words is related to their semantic similarity. Word embedding vectors can be obtained through some neural network-inspired algorithms, such as Word2Vec [31] and GloVe[1].

The Word2Vec algorithm [31] uses a neural network to learn word associations and represents each distinct word with a specific real number vector. This representation expresses similarity between words, as the proximity between vectors in the vector space represents the semantic similarity between words. The similarity between vectors is typically calculated using the cosine similarity, expressed as:

$$\text{sim}(a, b) = \cos(\theta) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{||\boldsymbol{a}|| \times ||\boldsymbol{b}||} \tag{2.12}$$

In the previous equation, $\boldsymbol{a}$ and $\boldsymbol{b}$ represent word vectors, and $\cos(\theta)$ represents the cosine of the angle between both vectors.

The Word2Vec algorithm has 2 fundamental models that can alternatively be used to compute the representations: the Continuous Bag of Words (CBOW) model and the Skip2Gram model. In brief, the CBOW model learns to predict a target word considering all words in its surrounding text. The sum of the context vectors is used to predict the target word, and the order of the context words does not impact the outcome. The number of words in the surrounding text that is taken into account is determined by a fixed window size. On the other hand, the Skip2Gram model weights the context words within a fixed window size, following the idea that a word closer to the target word has more impact to its representation than distant ones.

Alternatively, the GloVe algorithm [36] relies not only on local statistics, but also on the global statistics of a corpus to come up with word vectors. Training is performed on global co-occurrences counts of words matrix, which represents how frequently words co-occur with one another in a given corpus, with the objective of learning word vectors such that their dot product equals the logarithm of the words probability of co-occurrence.

Both these methods offer the ability to capture similarity between words, aside from tackling the

[1] https://nlp.stanford.edu/projects/glove/

problem of dimensionality that traditional text representation approaches have.

2.2.3 Contextual World Embeddings

Word embedding methods like Word2Vec lead to significant advances in NLP. Still, more recently, novel approaches for producing contextual embeddings have been achieving state-of-the-art performance on a wide range of NLP tasks [25]. Contextual embeddings methods assign to each word a representation based on its context, by that capturing many syntactic and semantic properties of words under diverse linguistic contexts. This contrasts with static embedding methods such as Word2Vec, where each word is always represented through the same vector, independently of the context surrounding the word. The most common models to generate contextual embeddings are the Embeddings from Language Models (ELMO) and the BERT methods.

The BERT model, described by [7], is inspired by other techniques including the ELMO model (based on LSTM's) and the Transformer that was previously explained.

BERT's model architecture corresponds to a multi-layer bidirectional Transformer encoder based on the original implementation. It is designed to train deep bidirectional representations by jointly conditioning on the left and right context in all layers, resulting in the capacity to be fine-tuned with the addition of an output layer to create models for downstream tasks (i.e., supervised-learning tasks that use a pre-trained model) such as classifying input texts.

There are two main steps involved in using BERT's framework: the pre-training phase and the fine-tuning phase. During the pre-training phase, the model is trained using unlabeled data over different tasks. During fine-tuning, BERT's representations are adjusted to a target task.

Unlike previous work, BERT is not designed to train left-to-right models or a concatenation of left-to-right and right-to-left models. It instead trains a bidirectional model that is believed to be strictly more powerful than its ancestors. However, one disadvantage of using this model is the fact that it allows the word to *see itself*, which makes the prediction a trivial problem. For that reason, the authors proposed a pre-training task that masks a percentage (15%) of input tokens and then predicts those masked tokens. This task is called Masked Language Modeling (MLM).

Many NLP downstream tasks are based on understanding the correlation between two sentences, which cannot be captured by a simple language modeling. One way to solve this is by pre-training the model for a task that predicts if the next sentence is the real next sentence of a previous one. This task is called Next Sentence Prediction (NSP). In detail, the model receives two sentences A and B as input, and decides if B is the actual next sentence of A. For 50% of the time, B is the actual next sentence of A and is labeled as positive (otherwise is labeled as negative).

The input to the BERT model can thus be a sequence of either one or two spans of text. The authors used WordPiece embeddings to represent the tokens associated with each word in the input

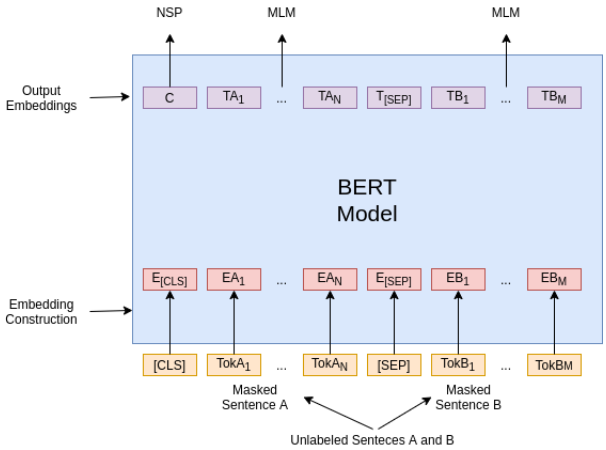


Figure 2.3: General structure of the BERT pre-training phase.

sequence. The first token of every sequence is a special classification token $[CLS]$. Aside from the token embeddings, it is also necessary to distinguish between sentences in a sequence. For that reason, there is an extra special token, $[SEP]$, that is used to separate sentences in a sequence. Also, and adding to token embeddings, there is a learned embedding for every token indicating if it belongs to sentence A or B. In short, the final embedding construction for the input is the sum of token embeddings, sentence segmentation embeddings, and position embeddings (i.e., a simple method to indicate the position of the token in the sequence). The drawback of using learned positional embeddings is that the input sequence is fixed and limited to the maximum sequence length defined at the pre-training phase, while the absolute encodings can be theoretically extended to larger input sequence lengths.

In Figure 2.3, a general representation of the BERT pre-training phase is illustrated. The input sentences are first masked and then tokenized (e.g., $TokA_1$ is the first word token of the sentence A). After that, tokens go through the embedding construction explained above (e.g., EA_1 is the embedding of the first word token of sentence A). Finally, the outputs generated by the model correspond to the final hidden vector of the special token $[CLS]$ as C, that is associated to the NSP prediction task, and the final hidden vector for each input token as T (e.g., TA_1 is the output for the token $TokA_1$).

During the fine-tuning phase, the model is initialized with the pre-training parameters obtained during the pre-training phase, and is then fine-tuned using labeled data from downstream tasks. At the output, the token representations are fed into an output layer for token level tasks, such as sequence tagging or question answering, and the C output is fed into an output layer for text classification tasks.

BERT has been the target of intense study in order to improve its objective and architecture. Since its creation, some variants have been developed. One of those variants is the RoBERTa model. RoBERTa makes a few changes to the BERT model and achieves meaningful improvements. The changes include: (1) training the model longer, with bigger batches and over more data; (2) removing the next sentence prediction objective; (3) training on longer sequences; and (4) dynamically changing the masking pattern

applied to the training data [26].

2.3 Overview

This chapter presented the fundamental concepts associated with the work developed in this dissertation. Section 2.1 explains fundamental concepts regarding neural networks and deep learning, together with strategies regarding the training of these models. In particular, deep neural networks architectures that are widely used in text classification are introduced, such as convolutional and recurrent neural networks as well as the Transformer model. Section 2.2 discusses methods to represent textual elements, from approaches based on one-to-hot representations to the most novel BERT pre-trained language model, passing through the Word2Vec algorithm.

3

Related Work

Contents

This chapter presents relevant related work covering the most critical areas. Section 3.1 describes different methods previously applied to the tasks of toponym resolution and document geocoding. Then, Section 3.2 presents previous work on general entity linking. Finally, Section 3.3 provides an overview on the approaches used by previously developed toponym resolution systems.

3.1 Toponym Resolution and Document Geocoding

The following subsections present relevant studies previously developed using different techniques, namely approaches based on heuristics (Section 3.1.1), methods that use language models (Section 3.1.2), and finally methods based on deep learning techniques (Section 3.1.3).

3.1.1 Methods Based on Heuristics

Most seminal toponym resolution methods are based on hand-built heuristics that rely on information gathered from metadata (e.g., population size, area, land coverage,etc.) present in gazetteers (i.e., geographical dictionaries containing information concerning the geographical makeup, social statistics, and physical/administrative geographic features of a country, region, or continent), to decide which real location is mentioned in the text. Multiple heuristics can be combined to create a more robust system. Still, while these systems tend to work well for common place names, they lack when dealing with uncommon or unknown places, which may not even be listed in gazetteers.

Systems based on heuristics rely on these hand-built rules to rank, within a set of candidates retrieved from a gazetteer, which candidate refers to the place name given in a span of text.

Leidner [19] compiled commonly used heuristics in diverse articles, and analyzed their results with new combined approaches also based on heuristics. This author also divided the heuristics into two subsets: external knowledge heuristics (i.e., heuristics relying on metadata from external resources) and linguistic knowledge heuristics (i.e., information gathered from the text itself).

This section will nonetheless consider a finer division, that splits heuristics into four sets, namely heuristics based on administrative divisions (i.e., a portion of a country or other region delineated for the purpose of administration), prominence heuristics, linguistic heuristics (following the concept presented by Leidner [19]) and geospatial heuristics.

Some heuristics based on administrative divisions that have been previously explored include:

- Superordinate mention ($H1$). Based on assigning a spatially contained toponym to the candidate located in the superordinate toponym related to the first one (e.g. assuming $t1$ to be resolved, and a second toponym $t2$ which can refer to a country $c2$), if it further holds that a candidate to t1 ($c1$) is located in $c2$, then assign $t1$ to $c1$;

- Singleton Capitals ($H2$). If a toponym only occurs once in a text, and has exactly one candidate that is a capital city, then assign it with that candidate;
- Prefer higher-level candidates ($H3$). If a toponym has two candidates, assign it with the one that belongs to a higher-level category in a taxonomy of administrative divisions, e.g. with the entire Earth as root and small cities as leaves;
- Feature type disambiguator ($H4$). If there is an indication of a feature type (e.g., city, capital, country, county, town, etc.) occurring together with the name of the candidate, then eliminate the candidates that are not of that type (or prefer the other ones);
- Spatial minimality ($H5$). Assume that if two toponyms are mentioned in a relatively small span of text, the smaller region is the one with higher impact in the assigning of candidates to both toponyms;

Some prominence heuristics that have been previously explored include:

- Prior probability on Wikipedia ($H6$). Assign a toponym with the candidate with the highest probability candidate from a database such as Wikipedia (i.e., the candidate that is most likely to be referred in a Wikipedia article);
- PageRank value on Wikipedia ($H7$). Assign a toponym with the candidate with the highest PageRank (i.e., an algorithm used to measure the importance of website pages) value on the Wikipedia link graph;
- Largest Population ($H8$). Assign a toponym with the candidate with largest population size. Alternatively, one can also prune the list of candidates;
- Default candidate ($H9$). Assign a toponym with the most known candidate based on existing knowledge (i.e., it relies on previous human intuitions about a given candidate, such as land coverage);
- Number of alternative names ($H10$). Assign a toponym with the candidate with more alternative names (i.e., distinct names to refer to one place name);

Some linguistic heuristics that have been previously explored include:

- One referent per discourse ($H11$). Assume that a toponym mentioned throughout the text corresponds to the same location in every case;
- Contained-in qualifier following ($H12$). Based on recognizing a local pattern and assigning the toponym to the spatially contained candidate (e.g., assuming $t1, t2$ as toponyms, and $c1, c2$ as candidates, and assuming that there is exactly one candidate $c1$ for $t1$ such that $c1$ is spatially contained in $c2$, where $c2$ is a candidate referent to $t2$, then assign that candidate $c1$ to $t1$);

- Frequency weighting ($H13$). Gives higher emphasis (i.e., weight) to more frequent toponyms in the text when applied with other heuristics;
- Textual similarity ($H14$). Assign a toponym with the candidate with the highest similarity value in textual similarity algorithms such as Levenshtein or Jaccard. This textual similarity score can be obtained by applying these algorithms on the place name itself or even descriptive texts;

Some geospatial heuristics previous explored are:

- Focus on geographic area ($H15$). Discard candidates outside of a given polygon by pruning the gazetteer, according to a given study region;
- Discard off-threshold ($H16$). Compute the centroid for the toponyms in the text and discard the candidates that are distant to that centroid (i.e., more than a defined standard deviation from it);
- Distance to unambiguous neighbors ($H17$). Assign a toponym $t1$ with the candidate closer to the surrounding unambiguous toponyms;
- Distance to unambiguous textual neighbors ($H18$). Assign a toponym $t1$ with the candidate that is closer to the surrounding unambiguous toponyms in the text;

Apart from compiling previously used heuristics, Leidner [19] also introduced a new algorithm (LSW03) based on two minimality language knowledge heuristics, $H5$ and $H11$ (in the previous list), assuming that textual proximity should be strongly correlated with spatial proximity. The author tested his system with another heuristic-based system named PERSEUS [42] that relied on heuristics $H3$, $H13$, $H16$ and $H17$ (in the previous list). The comparison was made on two known toponym resolution datasets: TR-CoNLL and TR-MUC4. Apart from comparing his approach with a complete system, he also compared it with baselined systems based on different individual heuristics.

When comparing both systems, Leidner [19] noticed that the performance is, in both cases, low, this can be due to the difficulty of the toponym resolution task on world-wide scope, where a resolver sometimes has to decide among hundreds of candidate referents. LSW03 achieved better recall results on both datasets, and better precision on the TR-CoNLL dataset. However, PERSEUS achieved better precision on the TR-MUC4 dataset. When comparing these results with the results achieved by the baseline approaches based on individualized heuristics, it can be said that a baseline system MAXPOP, that incorporates heuristic $H8$, is a very strong baseline that can be easily implemented. It was also concluded that the LSW03 system implementing heuristics $H5$ and $H11$ achieved better results than the baseline systems that implemented both heuristics individually, which indicates that systems that have several heuristics in combination might achieve better results.

More recently, and taking inspiration on general entity linking systems, several studies have considered supervised learning approaches that take heuristics as features in machine learning models. This way, the need to manually tune the weights given to each heuristic can be suppressed.

One of these studies was presented by Lieberman and Samet [22], introducing the concept of *adaptive context features* that considers a window of context around each toponym and uses geographic attributes of toponyms, in the window, to aid in their correct resolution. The *adaptive context features* are context-sensitive (i.e., they depend on the position of the toponym to be disambiguated in relation to other toponyms in the context window) and are divided into proximity features and sibling features. Proximity features assume that a candidate geographically located near the toponym to be resolved, as well as other toponyms in the window of context, is likely to be chosen (i.e., $H18$ in the previous list). On the other hand, sibling features assume that candidates that are nearby in the text, and share containers in a geographic hierarchy, are likely to be picked (i.e., $H5$ and $H12$ in the previous list). These features, along with other context-free features (i.e., features that do not rely on the position of the toponym in the text) such as the population of the candidate and number of alternative names for a candidate ($H8$ and $H10$, respectively), are then fed into a Random Forest classifier that decides, for each candidate, if it is correct or not. The system proposed by Lieberman and Samet [22] achieved better results (i.e., precision and recall) in the task of toponym resolution than other prominent competing methods, such as the Thomson Reuter's OpenCalais, and Yahoo!'s Placemaker, on widely used datasets, namely the Local Global Lexicon (LGL) corpus, the ACE corpus and the CLUST dataset.

Another approach, based on heuristics as features, was proposed by Santos et al. [39]. This system trained a Learning to Rank model to assign a rank to each candidate location according to the likelihood of it being the correct prediction.

More recently, in the context of the SemEval-2019 Task 12 challenge, some toponym resolution systems that take heuristic as features in meachine learning models were evaluated on scientific articles. Weissenbacher et al. [48] summarized the corpus used within this challenge (discussed later in this report), and the approaches presented by the teams.

Team DM_NLP achieved the best perfomance on all sub-tasks of the challenge [47], by dividing the system into two sub-tasks: toponym detection and toponym disambiguation. Regarding the toponym disambiguation sub-task, all possible candidate toponyms from GeoNames database are first retrieved. Second, a binary classifier with features as input is applied to each candidate to figure out whether it is the place that the mention refers to. More specifically, a candidate ranking score is calculated and the highest candidate is returned as the most likely candidate. Regarding input features, Wang et al. [47] divided them into four groups: name string similarity (i.e., $H14$ heuristic in the previous list), candidate attributes (i.e., $H9$ heuristic in the previous list), contextual features (i.e., $H14$ heuristic applied to descriptive Wikipedia texts of the candidates) and mention list features (i.e., take all mentions in a

document as a feature). The authors decided to use LightGBM [16] as their base model, since it usually has better performance than other gradient boosting models.

Another work that also achieved interesting results in the SemEval-2019 Task 12 Challenge is described in Li et al. [21]. A sub-task division into toponym detection and toponym disambiguation was also performed. Similarly to Wang et al. [47] work, candidate toponyms were generated from GeoNames database. However, instead of using a gradient boosting model, Li et al. [21] used an SVM classifier with hand-engineered features, such as: history result (i.e., if the toponym appears in the training set, history result refers to the ranking of the number of times the Geonames ID appears as a standard answer), population (i.e., $H8$ heuristic in the previous list), GeoNames feature codes (i.e., $H3$ heuristic in the previous list), name similarity (i.e., $H14$ heuristic in the previous list) and AncestorNames correlation. Li et al. [21] ranked 2nd in the toponym detection task, as well as in the toponym disambiguation task.

3.1.2 Methods Based on Language Models

In contrast to methods relying on gazetteers, some previous studies have shown that geolocation can be performed with accuracy sufficient for useful real-world application by using only the text of a document, even without any available metadata associated with the document. This is, for instance, the main thesis behind the work of Wing et al. [50]. Metadata-based approaches (like previously explored external-knowledge heuristics) can achieve good results but are very gazetteer dependent, making them not prone to generalization. On the other hand, text-based approaches can be utilized in any type of corpora.

In more detail, Wing et al. [50] presented some methods for supervised text-based geolocation that discretize the Earth surface into a grid of cells, each corresponding to a specific region, allowing the toponym resolution task to be dealt as a classification problem. The classification can be made through *language modeling approaches* (i.e., statistical models of the distribution of words in the document), or alternatively through discriminative approaches such as logistic regression.

The idea behind these methods is to map each training set toponym occurrence (or document, in the case of document geocoding) with the cell of the region associated with it. After that, all training instances associated with a specific cell are concatenated into a pseudo-document, and the language model of this so-called pseudo-document is calculated. The language of the test toponym occurrence is also computed and another method is then used to compare the two language models. The centroid of the highest-scoring cell is finally chosen as the predicted location for the document.

In particular, Wing et al. [50] explored different grid types for representing the Earth, such as uniform grids, in which rectangular cells of equal-size are created, and adaptive k-d tree grids, whose the main idea is to relate the size of the cell with the number of documents per cell. When the number of instances associated with a cell reaches a pre-defined threshold, the cell is split into two cells. The reader is encouraged to glance over Roller et al. [38] for further details regarding this Earth representation method.

In order to develop a proper language model (i.e., a method for assigning probabilities to sequences of words), some distributions were calculated, namely.

- In particular, the probability of a word j occurring in a document k can be expressed as:

$$\theta_{d_k j} = \frac{\text{count}(w_j, d_k)}{\sum_{w_l \in V} \text{count}(w_l, d_k)} \tag{3.1}$$

  In the previous expression, the numerator is the number of times the word j occurs in document k, and the denominator is the sum of the number of times each word l in the vocabulary occurs in document k.

  A Laplace smoothed distribution (i.e., even if a word is not seen in a document, a small value is assigned to it) can be computed through an interpolation with the global probability of a word occurring across all documents, introducing a discount factor, and can be expressed as:

$$\tilde{\theta_{d_k j}} = (1 - \lambda_{d_k} \theta_{d_k j}) + \lambda_{d_k} \theta_{D_j} \tag{3.2}$$

  In the previous expression, θ_{D_j} is the global distribution and λ_{d_k} indicates the discount factor, that can be assigned to a constant value (i.e., Jelinek smoothing) or can be calculated through the following formula (i.e., Dirichlet smoothing):

$$\lambda_{d_k} = 1 - \frac{|d_k|}{|d_k| + m} \tag{3.3}$$

  In the previous expression, $|d_k|$ is the size of the document and m is a tunable parameter.

- The probability of a word j occurring in a cell i can be expressed as;

$$\theta_{c_i j} = \frac{\sum_{d_k \in c_i} \text{count}(w_j, d_k)}{\sum_{d_k \in c_i} \sum_{w_l \in V} \text{count}(w_l, d_k)} \tag{3.4}$$

  In the previous equation, the numerator is the sum of times a word j occurs in a document (k) that is assigned to a cell i and the denominator is the sum of the number of times each word l in the vocabulary occurs in a document (k) assigned to the cell i.

- The probability of a cell i being the cell where the word j occurs is given by the following expression, where the numerator is the probability of a word j occurring in a cell i and the denominator is the probability of a word j occurring in every cell (i) of the grid;

$$k_{ji} = \frac{\theta_{c_i j}}{\sum_{c_i \in G} \theta_{c_i j}} \tag{3.5}$$

- The prior probability of a cell i is, in turn, given by the following expression, whose numerator is the number of documents assigned to cell i and the denominator is the amount of documents;

$$\gamma_i = \frac{|c_i|}{|D|} \tag{3.6}$$

After having a language model based on word and cell distributions, both for pseudo-documents and test instances, it is necessary to establish a method to compute the similarity between them. Information Retrieval approaches based on the Kullback-Leibler divergence (i.e., a measure of how one probability distribution is different from a second), or alternatively based on the standard Naïve Bayes classification model were considered but did not achieve good results. The following equations (3.7,3.8) describe these approaches, respectively.

$$\hat{c}_{KL} = \underset{c_i \in G}{\operatorname{argmin}} \sum_i \theta_{d_k} log \frac{\theta_{d_k}}{\theta_{c_i}} \tag{3.7}$$

$$\hat{c}_{NB} = \underset{c_i \in G}{\operatorname{argmax}} \gamma_i \prod_{w_j \in V_{d_k}} \theta_{c_i j}^{\#w_j, d_k} \tag{3.8}$$

One disadvantage regarding Naïve Bayes is that it assumes all features are independent, which penalizes models with many irrelevant features. For that reason, a feature selection method, namely the Information Gain Ratio, was implemented together with the original Naïve Bayes approach, achieving better results than the default attempt.

In addition to Information Retrieval models, discriminative classifiers such as a Logistic Regression model were also considered. However, one disadvantage of using a Logistic Regression model is that this approach is more time and resource-consuming, so a Flat Logistic Regression model was instead considered. A hierarchical classification approach implementing a multi-level discriminative model, was developed to suppress the performance limitations of the Logistic Regression model. To produce a hierarchy, a root cell c_{root} that spans the entire Earth is first created and, from there, a tree of cells at different scales is also created. This process is done recursively. A *local classifier per parent approach*, in which an independent classifier is learned for every node of the hierarchy above the leaves, is afterwards used. The probability of any node in the hierarchy is given by:

$$P(c_j) = P(c_j | \uparrow c_j) P(\uparrow c_j) \tag{3.9}$$

In the previous expression, the $\uparrow c_j$ is the parent of $c_j s$ in the hierarchy.

This approach allows the use of many classifiers, and each may have a different number of outcomes. It also lends itself to beam search (i.e., a heuristic search algorithm that explores a graph by expanding

the most promising node in a limited set), reducing the number of model evaluations to be performed.

Hierarchical classification achieved state-of-the-art results on well-known corpora (such as Twitter user feeds and Wikipedia articles), outperforming Naïve Bayes, Kullback-Leibler divergence and all other methods. Additionally, a Flat Logistic Regression model is able to beat Naïve Bayes on small corpora, but fails badly on large and worlwide corpora. Finally, feature selection through Information Gain Ratio is able to perform well on the Twiiter corpus, but underperforms on the Wikipedia corpus.

Wing et al. [50] also discussed the ties connecting toponym resolution and document geocoding, based on Speriosu and Baldridge [43]. The idea behind this approach is that toponyms are strong indicators of the location of the text. Toponym resolution can then aid document geocoding, by potentially disambiguate toponyms in the text, and by making use of external knowledge bases such as gazetteers.

Speriosu and Baldridge [43] explored an approach that considers non-toponym textual content for disambiguating toponyms, by creating a text-driven approach using indirect supervision. The training of their model was, indirectly, made through the learning of classifiers based on instances extracted from geo-tagged Wikipedia articles. If a toponym is at no more than a specified number of km from the document location, it is considered a mention of the document location. Speriosu [44] developed a number of methods for toponym resolution based on the idea previously presented by Speriosu and Baldridge [43]. It is worth mentioning that some of these methods rely on outside knowledge.

Wing et al. [50] decided to use a co-training strategy (i.e., a semi-supervised strategy introduced by Blum and Mitchell (1998) that allows for bootstrapping a classifier or other machine learning tool using only a small amount of labeled data and a large amount of unlabeled data) that involves a combination of document geolocation (previously explored in this report) and toponym resolution [44]. Co-training achieved noticeable increase in accuracy compared with a Naïve Bayes geolocator trained only on the base Wikipedia corpus, when both are evaluated on the CWar corpus.

Another study based on modeling the geographic distributions of words over the Earth's surface was reported by [4]. This approach (designed as *TopoCluster*) is motivated by the need to identify geographical clusters for each word in the vocabulary. These geographic likelihoods are derived by applying local spatial statistics over a large number of geo-referenced language models. Toponym resolution is performed by selecting the strongest overlapping point in the distribution for a specific toponym and its context surrounding.

TopoCluster uses the Local Getis-Ord G_i* statistic to measure the strength of association between words and geographic space. This statistic can be defined through the following equation:

$$G_i*(x) = \frac{\sum_{j=1}^{n} w_{ij} x_j}{\sum_{j=1}^{n} x_j} \tag{3.10}$$

In the previous equation, x_j is the measure of the strength of the variable x at the location j, and w_{ij} is the weight, given by an Epanichnikov kernel, that defines the similarity between locations i and j.

This kernel can be calculated as:

$$w_{ij} = .75 \left(1 - \left(\frac{\text{dist}(i,j)}{100km}\right)^2\right)_{\text{dist}(i,j) \leq 100km} \tag{3.11}$$

The result of the local statistic computations is a matrix with grid cells as columns and each word as a row vector, $g^*(x)$. The g_i* statistic is calculated from points in a regularized grid. The authors created a .5º geographic degree space grid, with more than 60.000 unique cells for representing the Earth.

Toponym disambiguation is performed by isolating toponyms (t) from non-toponyms (x) in a target toponym (z) context window (c) of 15 words. After that, the weighted sum of all the g^* values of the toponyms t and words x in c is calculated as:

$$g^*(z,c) = \theta_1 g^*(z) + \theta_2 \sum_{t \in c} g^*(t) + \theta_3 \sum_{x \in c} g^*(x) \tag{3.12}$$

The parameters θ_1, θ_2 and θ_3 weight the importance given to the toponym, to other toponyms in the context, and generic words in the context. The output of the system is the centroid coordinates of the cell i with the largest value in $g^*(z,c)$.

DeLozier et al. [4] also created a variation of the *TopoCluster* system relying on the use of a GeoNames-Natural Earth gazetteer, described as *TopoClusterGaz*. A reference for a toponym is matched to the gazetteer entry that matches the term and has geometry closest to the most overlapped point i. This system used a heuristic based on domain locality, wherein certain geographies are weighted according to their administrative division (i.e., $H3$ in the previously presented list).

TopoCluster performed well on three common toponym resolution datasets, namely TR-CoNLL, CWar, and LGL. When compared with end-to-end toponym resolution systems of Speriosu and Baldridge [43], *TopoCluster* achieved better results than any on the CWar dataset and was fairly competitive on the other datasets. The results of the gazetteer-dependent approach were similar to the gazetteer-free approach, indicating that gazetteer matching may not be that essential to create a competitive toponym resolution system.

3.1.3 Methods Based on Deep Neural Networks

In order to address the problem of toponym resolution, Gritta et al. [10] introduced the *CamCoder* system, which combines *MapVec*, a sparse vector representation that generates geographic features from text documents beyond lexical features (i.e geographic representation of location mentions), and *Lexical Geocoder* that uses representations based on lexical features.

The lexical part of the system has 3 inputs: the target entity (the place mention), other location mentions (other places mentioned in the same context), and context words (all words in a 200-word

window on each side except the target toponym). Each input is fed into separate layers that combine convolution and max-pooling operations, together with standard feedforward layers.

The *MapVec* also separate network components composed of two fully-connected feedforward layers. Finally, all layers are connected into a softmax layer predicting one of 7823 classes, each being a 2x2 degree tile representing part of the world's surface.

MapVec is initialized as a world map of 1ºx1º geodesic tiles which is then populated with the prior geographic distribution of each location. This distribution is calculated using population counts, since more populous places are more likely to be mentioned in a textual reference. The exact procedure is given in Algorithm 3.1. Finally, the world map is reshaped into a 1D vector.

Algorithm 3.1: *MapVec* generation

begin
  $Locs \longleftarrow extractLocations(Text)$
  $MapVec \longleftarrow newarray(length = 23,002)$
  for $each\ l\ in\ Locs$ **do**
    $Cands \longleftarrow queryCandidatesFromDB(l)$
    $maxPop \longleftarrow maxPopulationOf(Cands)$
    for $each\ c\ in\ Cands$ **do**
      $prior \longleftarrow populationOf(c)/maxPop$
      $i \longleftarrow coordinatesToIndex(c)$
      $MapVec[i] \longleftarrow MapVec[i] + prior$
  $m \longleftarrow max(MapVec)$
  return $MapVec/(m)$

CamCoder achieved state-of-the-art results, outperforming other geolocation systems, including heuristic based, language model based and machine learning based geocoders. Another key result obtained is the fact that the language model baseline achieved the best results among language-only geocoders. *MapVec* also remains effective when used together with other feature-based frameworks (CNN's, Random Forests, and MLP's), and improves accuracy when combined with other neural models, such as LSTM's.

Another recent deep learning approach based on contextual embeddings was developed by Cardoso et al. [3]. This system relies on contextual embedding models such as ELMO or BERT to transform the input text, feeding these representations into a neural network in order to make a region classification based on a geodesic grid.

Unlike Gritta et al. [10], Cardoso et al. [3] decided to use a geodesic grid based on the HEALPix scheme [9]. This algorithm partitions a spherical representation for the Earth's surface, generating equal-area cells corresponding to different Earth regions. The number of generated cells is calculated through a fixed number representing the desired resolution.

The system has three inputs: the toponym to be disambiguated, the words around the toponym (i.e.,

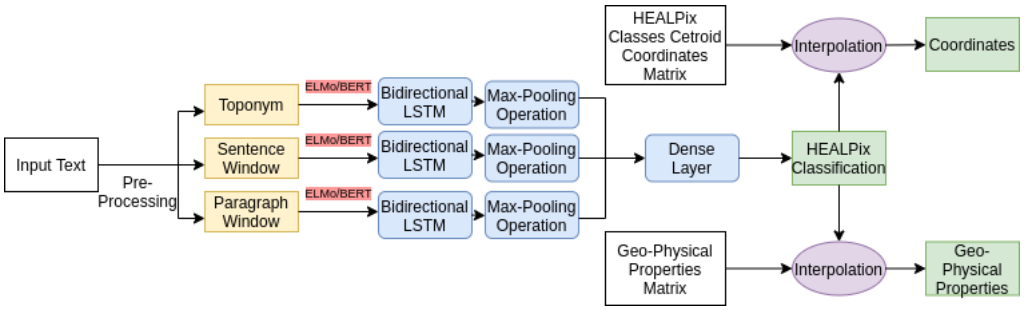


Figure 3.1: Cardoso et al. [3] model general architecture.

a fixed window to the left and right sides of the toponym, resulting in 50 words), and a paragraph of text (i.e., a fixed window with larger dimensions, 500 in this case). Each of the three inputs is then transformed into a sequence vector by going through a contextual embedding model. The authors tried both the ELMO and BERT models for this purpose.

Each of the resulting sequences of vectors (i.e., one for each word piece) is then fed into a custom bi-directional LSTM. The result of each unit is used as input to a max-pooling operation layer, and the resulting three vectors are then concatenated into a single vector representing all three inputs. The single vector representing the inputs is processed by a fully-connected layer that predicts HEALPix regions through a softmax activation function. The vector corresponding to the class probabilities is one of the model's outputs, which means that a loss function (i.e., cross-entropy, in this case) is computed over this result.

However, the coordinates of the predicted region need to be computed as the main output. For this purpose, the authors used the HEALPix class probability vector to interpolate the coordinates within a matrix representing the centroids of each HEALPix class, generating coordinates that are then connected to another loss function that computes the great circle distance between the predicted and the real coordinates.

The proposed system achieved state-of-the-art results on three different datasets: War of the Rebellion (WOTR), LGL, and SpatialML, when compared with other systems such as the *TopoCluster* and the *CamCoder*, previously explored in this section. The authors also tested the increase of training data by collecting more instances from Wikipedia, resulting in slightly better performance, although not consistently. Slightly better results were also achieved when using BERT instead of ELMO for embedding the input to the model.

More recently, Radford [37] presented yet another neural network-based method named ELECTROmap. This end-to-end probabilistic model for toponym resolution relies on the fine-tuning of a transformer language model, namely the DistillRoBERTa, to minimize the negative log-likelihood of a five component mixture of von Mises-Fisher (vMF) distributions.

More particularly, the vMF distribution generalizes the von Mises distributon beyond two dimension

to the surfaces of sphres or hyperspheres. The vMF probability density function is given by the following equation, where μ represents the mean direction, κ the concentration parameter, p the dimension considered, and x a point in p space:

$$f_{vmf}(x;\mu,\kappa) = \frac{\kappa^{p/2-1}}{(2\pi)^{p/2} I_{p/2-1}(\kappa)} e^{(\kappa\mu^T x)} \tag{3.13}$$

For every input text x_i, the model predicts parameters for five parameters vMF distributions, as well as a set of mixing probabilities describing the weights given to each of the five components. By using five components, the model can then fit a more flexible distributional shape than it would be able to with a single vMF component. The loss function of the ELECTRO-map model is given by the following equation, where ρ represent the mixing probabilities assigned to each component:

$$\begin{aligned} -L(\boldsymbol{W}) &= -\sum_{k=1}^{5} \rho_k \sum_i \ln(f_{vmf}(\boldsymbol{y}_i; \hat{\mu}_{ik}, \hat{\kappa}_{ik})) \\ \hat{\mu}_i &= f_\mu(\boldsymbol{x}_i; \boldsymbol{W}) \\ \hat{\kappa}_i &= f_\kappa(\boldsymbol{x}_i; \boldsymbol{W}) \end{aligned} \tag{3.14}$$

The authors also proposed several solutions for aggregating results to a single latitude/longitude prediction per observation. The first and more feasible solution is to take the single highest probability prediction. The second solution requires a priori knowledge, in a sense that the best prediction from the mixture is taken. Finally, the third solution takes a random prediction from the mixture. The ELECTRO-map system achieved state-of-the-art results while choosing the first solution to aggregate results, when compared to Mordecai (i.e., a full-text geoparsing system that extracts place names from text, resolves them to their correct entries in a gazetteer, and returns structured geographic information for the resolved place name) [12].

Another system developed for toponym resolution is the Multi-Level Geocoder (MLG) proposed by Kulkarni et al. [18]. Unlike *CamCoder*, this system does not rely on gazetteer metadata and population signals, thus avoiding biased predictions towards locations with large populations.

MLG is a CNN-based text-to-location system that learns spatial language representations by mapping toponyms from text to coordinates on Earth's surface. Locations are represented using partitions obtained through the S2 geometry library (i.e., the surface of the Earth is projected onto six faces of a cube and each face is hierarchically subdivided forming S2 cells in a quad-tree) supporting multi-level representation from a coarser to a finer granularity.

MLG resembles the *CamCoder* system in that it has the same three inputs, and each input is fed into separate layers that join convolution and pooling operations. However, MLG instead uses GloVe embeddings as input to the model, and uses multi-level S2 cells as the output of a multi-headed feature

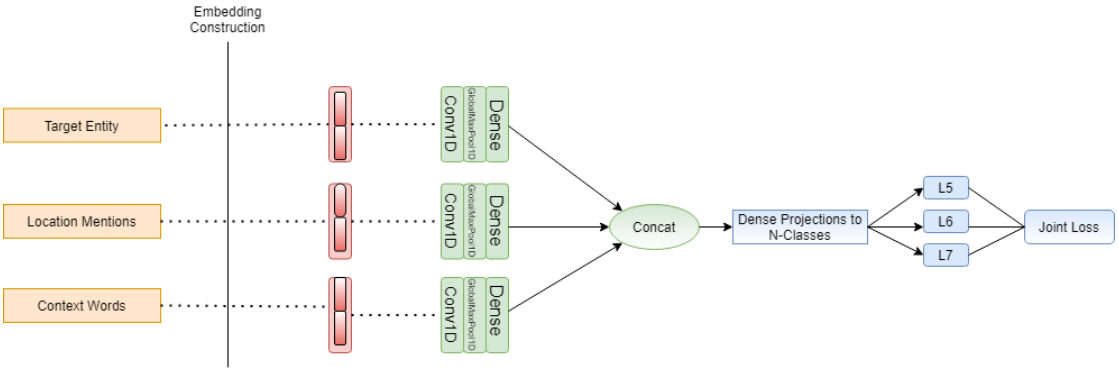


Figure 3.2: MLG model general architecture.

encoding model (since the penultimate layer maps representations of the input to probabilities over S2 cells). In particular, the system focuses on three different levels: L5 (83km^2), L6 (21km^2) and L7 (5km^2), each giving 6k, 24k, and 98k output classes, respectively. The model defines losses at each level and minimizes them jointly. Figure 3.2 shows the general architecture of the MLG model.

The authors evaluated their system in the same datasets used to evaluate the *CamCoder* system. Additionally, the authors also verified inconsistencies in the true coordinates of the different datasets, whereby decided to unify the true coordinates corresponding to the same target entity, thus creating a consistent evaluation. MLG results were compared with the *CamCoder* system, and a variant of the MLG that only uses a single layer (in this case L7), named SLG. MLG achieved better results than its competitors, displaying that the architecture inherited from the *CamCoder* system is effective on text geocoding. Moreover, results also showed that it is possible and even preferable to solely rely on lexical clues present in the text.

3.2 General Entity Linking Methods Based on Deep Neural Networks

Entity Linking (EL) is the task of identifying an entity mention in text and establishing a link to an entry in a structured Knowledge Based (KB), such as Wikipedia [40]. This is an essential step in many natural language processing and information extraction pipelines since it resolves ambiguity problems associated with named entities. Toponym resolution can then be seen as a specific type of EL.

Usually, the entity process consists of two sub-tasks: Entity Recognition (ER) and Entity Disambiguation (ED). ER is the process of distinguishing entity mentions in the text, while ED is the process of assigning an entity to those mentions.

As in many other NLP tasks, neural networks have proven to produce state-of-the-art results in the EL task. Neural entity linking models rely on entity vector representations and entity encoding methods,

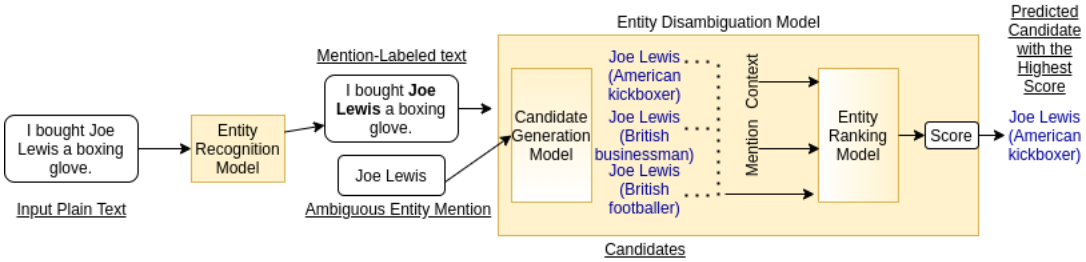


Figure 3.3: Example of a common entity linking architecture.

since it has been proven that encoding the KB structure and entity definitions helps to improve the generalization effectiveness of EL models.

The common architecture for neural entity linking includes treating the process as a ranking problem, including 3 important steps: identify the mention boundaries in text, produce a list of possible entities (candidates) for a given mention, and estimate how well a candidate entity matches the given mention. Figure 3.3 illustrates these steps.

The first step usually includes the use of named entity recognition techniques, that detect a word or set of words that form an entity. Alternatively to the use of standard supervised models for named entity recognition, one can use parsing outputs to detect noun phrases, or simply consider all word n-grams up to a given length.

The second step is referred to as candidate generation and its goal is to provide a list of possible senses for a mention, as specified by entities in a KB. This step is of extreme importance since it reduces the number of entities a mention can be referring to, by doing this preliminary filtering. There are three solutions for this problem: a surface from matching, a dictionary lookup, and a prior probability computation. The first method consists of a composition of a list of candidates that match surface forms of mentions in the text. The second method constructs a dictionary of additional aliases using KB metadata, like disambiguation pages from Wikipedia. In the third approach, the candidates are generated based on precalculated prior probabilities of correspondence between certain mentions and entities. These prior probabilities can be computed using count statistics usually relying on Wikipedia anchor links.

Given the list of candidates, the final step will compute a score for each one of them. For doing this, it is common to capture information from the context of the mention, and thus it is necessary to generate a contextualized vector representation of the mention, usually by an encoder network. Several different models using convolutional encoders and attention between candidate embeddings and embeddings of context words have been proposed. However, in recent studies, recurrent networks and self-attention techniques have been achieving the best results. Different architectures using LSTM cells have been adopted to EL, whether having one network for left context words and one network for right context words, or using a bi-directional network on context words represented through embeddings [17, 30].

Pre-trained language models such as ELMO and BERT have also been adopted for context and mention encoding [41]. More recent models have also been using self-attention usually relying on outputs from pre-trained BERT layers [51, 52].

Apart from context mention representation, there is also a need to encode candidate entity representations. There are several ways to encode entities, and some methods adapt word representations from the Word2Vec [31] model, either by extending the objective function with a joint alignment function based on several features of entities, therefore requiring sparse entity-entity statistics (although this issue can be resolved by giving another formulation of the objective function based on entity-word statistics), or simply by replacing the raw text with entity annotated text without any additional statistic.

More recently, some models relying on annotated data have been capturing entity information based on entity features such as entity type, description page, linked mention, title, or category information. These neural architectures for learning entity representations are designed to keep mentions and entities in joint semantic spaces, although allowing them to be correlated when doing a ranking system through vector similarity. Three common score functions used to train these neural models are the similarity score of entity and mention representation (which is the most widely used function), alignment score functions, where several features are learned independently and then merged, and matching functions relying on cross or bi-encoders for entities and mentions, built over the BERT architecture.

Most recent studies compare these representations (i.e., context mention representation and candidate entity representation) using dot product or cosine similarity. The calculated similarity score is often combined with mention-entity priors obtained during the candidate generation step, or with other features depending on string matching of entity types. The final choice is made using a probability distribution, obtained by a softmax function over the candidates. Figure 3.4 describes a general entity ranking model architecture as described by Sevgili et al. [40].

Finally, there is also the problem of entities missing in the KB's. The EL system must be capable of predicting the absence of a reference, known as NIL prediction. There are several ways to deal with this problem. When the candidate generation does not generate any candidate for a given mention, this mention is defined as an unlinkable mention. On top of this, one can add a score threshold below which mention is considered unlinkable. Some models add a NIL entity in the ranking phase, which makes it possible for it to be predicted as the best match for a given mention. Other models train an additional binary classifier that has as input mention-entity pairs, described by meaningful features, and decides whether the mention is unlinkable or not.

One of the most recent approaches, presented by Mulang et al. [33], uses WikiData as the KB and consists of a pipeline of two attentive LSTM-based neural networks for solving EL tasks. WikiData offers unique characteristics, by being collaboratively edited and by having no strict naming convention nor a standardized approach. This adds aliases to entities and their description as entities properties (e.g.,

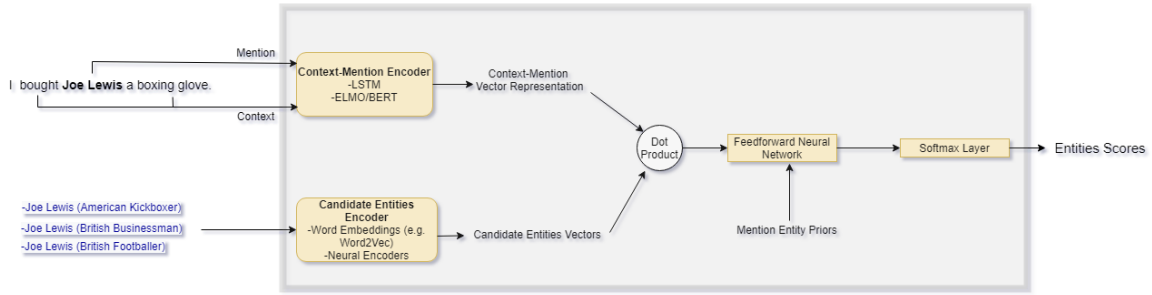


Figure 3.4: A general entity ranking architecture.

Obama and Barack Hussein Obama are aliases for the entity Barack Obama), which can be extremely useful for EL tasks. The authors analyzed the impact of this additional context information on attentive neural networks.

The system from Mulang et al. [33], named Arjun was developed to link entities by exploring the unique characteristics of WikiData. Having aliases of entities might help a neural network understand the context of the potential candidates. The authors extracted all WikiData entity labels and aliases as indexed documents in a local background knowledge graph (Local KG). As in most EL models, Arjun performs three sub-tasks: mention extraction, candidate entity generation, and entity ranking, which selects the most appropriate candidate entity for the given mention. For the first and third tasks, this method adopts a neural model inspired by the work of Luong et al. [28], that uses a bidirectional LSTM for the encoder and a one-directional LSTM for the decoder The input of the decoder is the source text sequence that is encoded and then unfolded into a target sequence by the decoder. An attention layer was also added to improve the model performance. Instead of encoding the input sequence into a single fixed context vector, the attention model develops a context vector that is filtered specifically for each output time step. In the first task, the model receives the source text sequence as input and identifies the mentions present in the text. In the entity generation step, each entity for which entity labels in the Local KG exactly match the mentions produced in the previous step is retrieved. Finally, a second model (similar to the one used in the first task) receives the full list of candidates and predicts the best WikiData entity labels, i.e. a text sequence is fed into the encoder as the input sequence, where the context of the text sequence is decided by using extra information in the form of associated alias.

The authors decided to not compare their results to generic entity linking approaches, which typically either do not use any background knowledge or employ other KGs. Instead, Arjun was compared with a baseline model, which performs the entity recognition task using an attentive neural network without any background knowledge (i.e., it maps the text to a sequence of WikiData entities without identifying surface form candidates), and another WikiData end-to-end EL model named OpenTapioca [6]. It was observed that Arjun achieved an 8% improvement in the performance over the baseline model, and that it significantly outperforms OpenTapioca.

3.3 Overview

Previous Studies	Heuristics																	Combining Heuristics through Supervised Learning	Language Models	Deep Learning
	Administrative division					Prominence					Linguistic				Geospatial					
	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11	H12	H13	H14	H15	H16	H17	H18		
Smith and Crane [42]			✓									✓	✓			✓		✓		
Amitay et al. [1]								✓			✓						✓	✓		
Leidner [19]					✓						✓	✓						✓		
Grover et al. [11]				✓				✓				✓					✓			
Lieberman et al. [24]				✓				✓				✓					✓			
Lieberman and Samet [22]					✓			✓	✓	✓		✓						✓	✓	
Karimzadeh et al. [15]			✓					✓						✓						
Zhang and Gelernter [53]				✓				✓	✓	✓		✓		✓			✓		✓	
Santos et al. [39]						✓	✓	✓	✓	✓			✓	✓			✓	✓	✓	
Weissenbacher et al. [49]	✓							✓			✓						✓			
Li et al. [21]			✓					✓						✓					✓	
Wang et al. [47]									✓					✓					✓	
Speriosu and Baldridge [43]																			✓	
Wing et al. [50]																			✓	
DeLozier et al. [4]																			✓	
Gritta et al. [10]								✓												✓
Cardoso et al. [3]																				✓
Radford [37]																				✓
Kulkarni et al. [18]																				✓

Table 3.1: Summary of techniques used in previous toponym resolution systems.

4

Toponym Resolution in Text

Contents

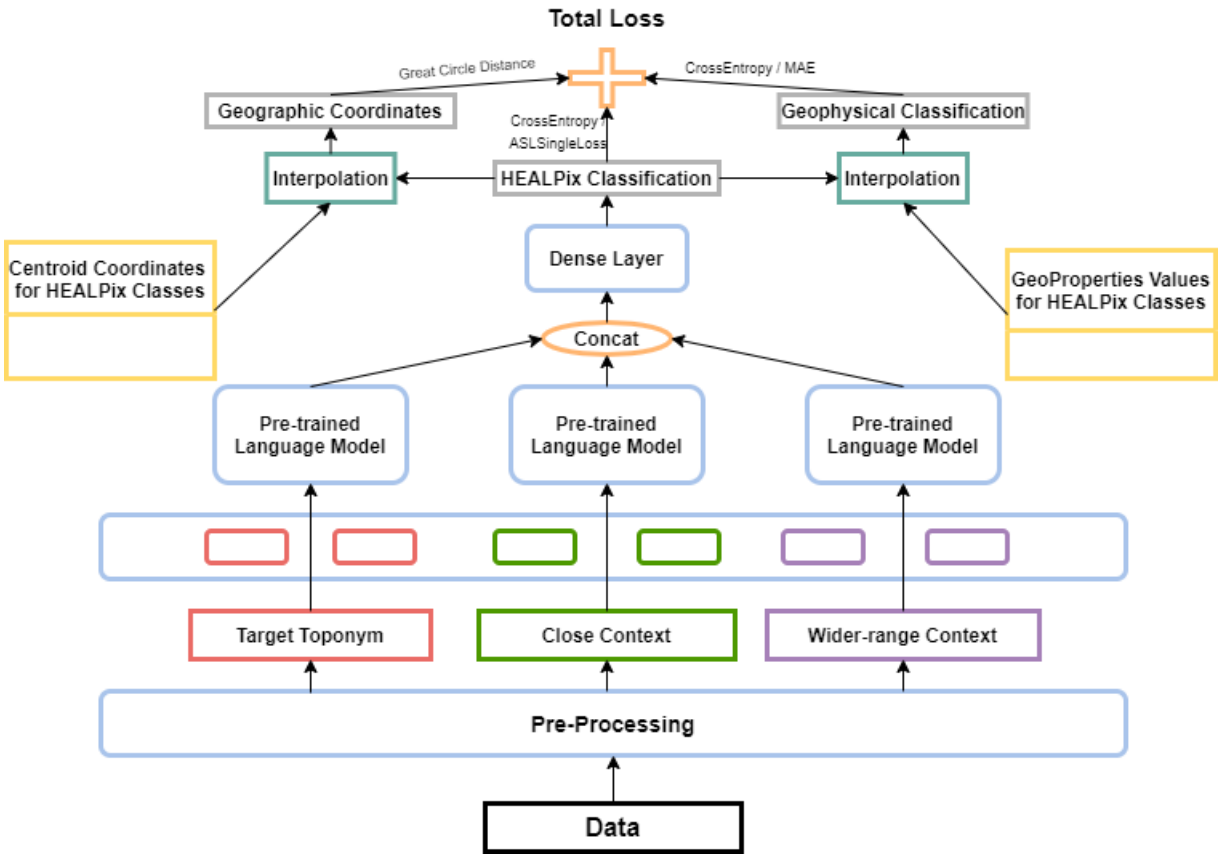


Figure 4.1: Proposed Model architecture.

The end-to-end toponym resolution model (Figure 4.1) detailed in this article relies on the fine-tuning of a pre-trained language model, and builds a representation from the toponym that is to be disambiguated, as well as its surrounding context. This section is organized as follows: Section 4.1 describes the architecture of the model, while Section 4.2 details the use of geophysical properties as an additional prediction in my model. Section 4.3 illustrates the training hyperparameters used in the experiments. Finally, Section 4.4 provides an overview of the topics discussed in this chapter.

4.1 Proposed Model Architecture

The proposed model considers multiple textual inputs, following the input partition proposed by Cardoso et al. [3]. Firstly, the textual document needs to be tokenized (i.e., splitting strings in sub-word token strings), using the language model tokenizer. Then, the tokenized text is divided into three sequences of tokens, namely: the toponym mention to be disambiguated (target toponym), the close context around the toponym mention (i.e., a fixed window, to the left and right sides of the target toponym, totalling 50 tokens), and a wider range context around the toponym mention (i.e., a fixed window, to the left and right

sides of the target toponym, totalling 510 tokens). It is worth mentioning that most language models have a maximum length limit for the input of 512 tokens, and knowing that each input vector needs to be fed with a [CLS] and a [SEP] token (at the beginning and ending of the vector, respectively), the bare maximum of actual textual input tokens has to be 510 (see Section 2.2.3).

Each of these sequences is then fed into a pre-trained language model, sharing parameters for the processing of the three sequences. Since we are dealing with a classification task, only the first output vector associated with the special token [CLS] is extracted from each. The resulting three vectors are then concatenated to form a representation of the whole input data. This vector is then processed by an output layer that produces a probability distribution over the regions that were defined by the Earth partitioning algorithm.

As previously mentioned, I choose to tackle the toponym resolution task as a prediction problem, where each place name reference is associated with a certain region of the Earth through a geodesic grid. I then use the classification probability distribution to obtain geographical coordinates (i.e., latitude and longitude) of each recognized place name, through a regression loss.

The geodesic grid used to support the classification is built through the HEALPix scheme, proposed by Gorski et al. [9]. In brief, the HEALPix algorithm partitions a spherical representation for the Earth's surface, generating equal are cells corresponding to different regions. Throughout the experiments reported on this article, I fixed the resolution parameter to N_{side}=256, which is equivalent to considering a maximum of 786,432 regions. However, in reality, the number of classes will be much smaller, given that most regions will not be associated to any instance in the data.

The predicted class probability over the partition schema is one of the outputs of the system, so a loss function is computed over it. A natural choice is to compute the cross-entropy loss between the predicted probabilities and the one-hot true vector class. Alternatively, and taking advantage of studies performed in the field of Computer Vision, I also test the use of an asymmetric loss function. Asymmetric loss functions were first introduced by Zhou et al. [54], and later explored by Ben-Baruch et al. [2], and are robust to learning with noisy labels for various types of noise.

The predicted class probability is then passed through a softmax layer. The softmax probability distribution vector is then raised to the third power and re-normalized (i.e., a more peaked distribution is built from the softmax results, emphasizing the most likely region), and a final interpolation between this peaked probability distribution vector and a vector representation of the cell centroids (i.e., a matrix that contains the centroid coordinates of each HEALPix class) is performed to obtain the predicted geospatial coordinates of the toponym to be disambiguated, similarly to what was done in the work of Cardoso et al. [3]. The obtained vector is then connected to a new loss function that computes the great circle distance between the predicted and the ground-truth coordinates.

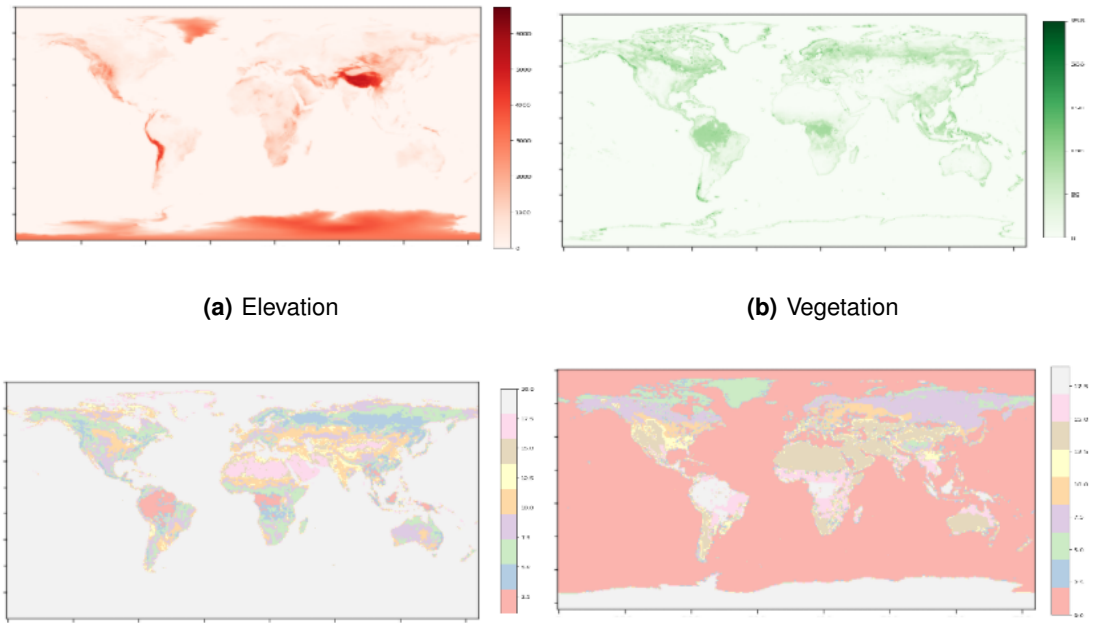
Class	Modern Land Coverage	Historic Land Coverage
0	——————————————-	Oceans/Water
1	Broadleaf Evergreen Forest	Cultivated Land
2	Broadleaf Deciduous Forest	Pasture/land used for grazing
3	Needleleaf Evergreen Forest	———————————————
4	Needleleaf Deciduous Forest	———————————————
5	Mixed Forest	Ice
6	Tree Open	Tundra
7	Shrub	Wooded tundra
8	Herbaceous	Boreal forest
9	Herbaceous with Sparse Tree/Shrub	Cool conifer forest
10	Sparse vegetation	Temperate mixed forest
11	Cropland	Temperate deciduous forest
12	Paddy field	Warm mixed forest
13	Cropland / Other Vegetation Mosaic	Grassland/Steppe
14	Mangrove	Hot desert
15	Wetland	Scrubland
16	Bare area, consolidated (grave, rock)	Savanna
17	Bare are, unconsolidated (sand)	Tropical woodland
18	Urban	Tropical forest
19	Snow/Ice	No data over land (e.g. Antarctica)
20	Water bodies	———————————————

Table 4.1: Land coverage classes

4.2 Geophysical Properties

Additionally, and besides HEALPix regions and geospatial coordinates, other outputs can be considered. In order to guide the model towards more accurate and informed location predictions, geophysical properties, such as land cover, elevation, and percentage of vegetation, associated with the predicted HEALPix cell, are also estimated.

This geophysical information is extracted from datasets in the raster format (i.e., a matrix of cells organized into rows and columns where each cell contains a value representing information), and incorporated into the model with the same interpolation technique used to estimate the prediction of geospatial coordinates, previously described. More specifically, each of the geophysical properties is encoded into real values, and then vectors corresponding to the measurements associated to each HEALPix class are created. In order to obtain these values, a polygon associated with the HEALPix class is used to crop the original rasters, thus obtaining a representative value of the region, and not only of the HEALPix cell centroid. The dot product between each vector and the previously calculated peaked probability distribution vector is computed, and the results are connected to new loss functions that correspond to the absolute difference between the predicted and the real values (in the case of the elevation and vegetation geophysical properties), and the cross-entropy between the predicted and label vectors (in the case of the land coverage geophysical property).



(a) Elevation

(b) Vegetation

(c) Land Coverage (2010's)

(d) Land Coverage (1850's)

Figure 4.2: Geophysical properties geographical distribution .

The raster datasets are obtained from the "Global Map data archives" project, and the "International Satellite Land-Surface Climatology Project" initiative. The following geophysical properties are considered: (1) the land coverage classification (i.e., amount of developed versus natural terrain), inferred from an historical source[1] (in the case of the WOTR corpus), and from modern sources[2] (in the remaining datasets); (2) the percentage of vegetation[3] (i.e., the ratio of the area covered with branches and leaves of trees); (3) the terrain elevation[4] (i.e., the elevation data at 1m interval covering the whole world).

Figure 4.2 illustrates the geographical distribution of geophysical properties considered. Regarding the land coverage images, one should also consider Table 4.1 that details the code classes for each of the considered land coverage rasters (i.e., the modern and the historic land coverage rasters).

4.3 Training

My model is trained using PyTorch [35], by using the pre-trained language model architectures provided by the transformers library [5]. The model is trained for 6 epochs, using a optimal learning rate of 0.0003,

[1] https://thredds.daac.ornl.gov/thredds/catalog/ornldaac/967/historic_landcover_hdeg/catalog.html?dataset=967/historic_landcover_hdeg/historic_landcover_hd_1850.nc4

[2] https://globalmaps.github.io/glcnmo.html

[3] https://globalmaps.github.io/ptc.html

[4] https://globalmaps.github.io/el.html

[5] https://huggingface.co/transformers/

and AdamW as optimizer [27]. The learning rate scheduler used creates a schedule with a learning rate that decreases linearly from the initial learning rate set in the optimizer to 0. Each training batch has a total of 8 examples. Given the fact that some versions of these pre-trained language models are usually extremely big and require major computational resources to fine-tune and even produce inferences, I consider a smaller batch size for the large versions of the language models (this is explained later on the document). In particular, a gradient accumulation technique (i.e., running a configured number of steps without updating the model variables while accumulating the gradients of those steps and then using the accumulated gradients to compute the variable updates) is used to simulate the original batch size of 8. Additionally, I use the Healpy python library[6], based on the HEALPix scheme, to generate the regions over the Earth's surface.

The cost function of my model can be defined as a combination of the following loss functions:

- (1) HEALPix region classification. One can either consider the cross-entropy loss or the asymmetric loss between the predicted class probabilities and the one-hot true vector class (i.e., labels).

$$\begin{gathered}\text{regionClassification}(outputs, labels) = \text{crossEntropyLoss}(outputs, labels) \\ \text{or} \\ \text{regionClassification}(outputs, labels) = \text{ASLSingleLabelLoss}(outputs, labels)\end{gathered} \tag{4.1}$$

- (2) Coordinates prediction. Great circle distance computed between the predicted coordinates and the ground-truth ones.

$$\text{greatCircleDistance}(\phi_1, \lambda_1, \phi_2, \lambda_2) = \sin\left(\frac{\phi_2 - \phi_1}{2}\right)^2 + \cos\phi_1 \cos\phi_2 \sin\left(\frac{\lambda_2 - \lambda_1}{2}\right)^2 \tag{4.2}$$

  Where ϕ_1 and λ_1 represent the latitude and longitude values of the predicted coordinates, and ϕ_2 and λ_2 the latitude and longitude values of the ground-truth coordinates.

- (3) Geophysical properties predictions. The mean absolute error is computed between the predicted values and the ground-truth values for the elevation and vegetation geophysical properties. Regarding the land coverage property, one can consider the cross-entropy loss between the predicted class probability and the one-hot true vector class.

$$\text{elevationLoss}(ePred, eLabel) = \text{MAE}(ePred, eLabel) \tag{4.3}$$

$$\text{vegetationLoss}(vPred, vLabel) = \text{MAE}(vPred, vLabel) \tag{4.4}$$

$$\text{landCoverageLoss}(lcPred, lcLabel) = \text{crossEntropyLoss}(lcPred, lcLabel) \tag{4.5}$$

[6] https://pypi.org/project/healpy/

It is worth mentioning that given the fact that each of the previously described loss functions produce different range values, the contribution of each loss function to the final combined loss is weighted (i.e., different weights to each loss function were tested, and the ones achieving the best results were kept). Model training therefore involves minimizing the combined loss functions associated to each of the outputs. The following equation resumes the cost function of my model where γ_n represent the result of the nth previously described equation.

$$\text{costFunction} = 1 * \gamma_1 + 0.005 * \gamma_2 + 0.1 * \gamma_3 + 0.1 * \gamma_4 + 0.01 * \gamma_5 \tag{4.6}$$

4.4 Overview

This chapter introduced the neural model proposed to address the task of toponym resolution. Section 4.1 presented the proposed model architecture and the geodesic grid used to support the classification. Section 4.2 presented the use of geophysical properties as an additional source of information for the model. Finally, Section 4.3 presented the training hyperparameters and the different terms of the cost function used in my experiments.

5

Experimental Evaluation

Contents

Statistic	SpatialML	LGL	WOTR	SemEval-19 Task-12 Corpus	GeoVirus
Number of Documents	428	588	1644	105	229
Number of Toponyms	4783	4462	10377	4659	2167
Number of Distinct Toponyms	825	1201	1970	950	685
Average Number of Toponyms per Document	11.18	7.59	6.31	44.37	9.46
Number of Distinct HEALPix classes	461	758	999	801	588
Number of Train Examples	4145	4016	8458	4193	1564
Number of Test Examples	638	446	1919	534	601

Table 5.1: Statistical characterization of the used corpora

This section describes the overview of the experiments conducted. Section 5.1 characterizes the corpora used throughout the experiments. Section 5.2 illustrates the methodology used to perform the experiments, and Section 5.3 provides the results of those experiments. Finally, Section 5.4 provides an overview of the topics discussed in this chapter.

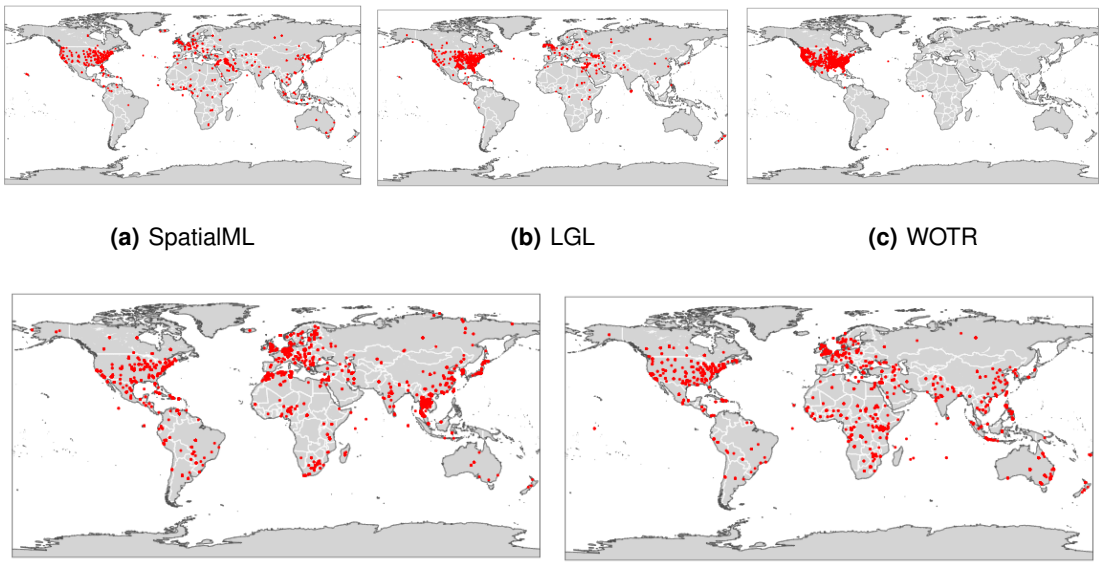
5.1 Corpora Used in the Experiments

I use four well-known public datasets: WOTR [5] [1], the LGL [23][2], the SpatialML [29] and the GeoVirus [10][3]. These corpus have been target of intense study over the years in the area [3, 4, 5, 10, 39]. Additionally, the training corpus used in the context of the SemEval-2019 Task-12 Challenge [48] [4] is also used in my experiments.

The SpatialML corpus is a subset of the ACE 2005 English SpatialML Annotations [29], available from the Linguistic Data Consortium. It contains documents that represent a variety of data sources, among which are broadcast news, magazine news, and web blogs. Each document is annotated using an XML-based language also called SpatialML, that allows the association of toponyms in the text with their respective locations and other geographically-relevant attributes. It should nonetheless be noted that,the SpatialML corpus is limited to the purpose of evaluating local toponyms, since news are usually directed to a much more geographically distributed audience. For that reason, Lieberman et al. [23] presented the LGL corpus, composed of articles from a variety of smaller distributed geographic newspapers. This corpus was specifically created for challenging toponym resolution systems, as it contains highly ambiguous names, including small cities and local mentions.

DeLozier et al. [5] proposed another corpus, in this case composed of annotated documents from a set of American Civil War archives, known as War of the Rebellion. Some of these archives contained military reports and orders, and others contained historical correspondence. It was concluded that WOTR was the most challenging corpus at the time (i.e., end-to-end toponym resolution systems achieved lower performance than in other previous developed corpora, such as LGL), as it contained

[1] https://github.com/utcompling/WarOfTheRebellion
[2] https://github.com/milangritta/Pragmatic-Guide-to-Geoparsing-Evaluation
[3] https://github.com/milangritta/Geocoding-with-Map-Vector/tree/master/data
[4] https://competitions.codalab.org/competitions/19948



(a) SpatialML **(b)** LGL **(c)** WOTR

(d) SemEval-2019 Task 12 Corpus **(e)** GeoVirus

Figure 5.1: Geographic distribution of toponyms in each dataset.

place names not in gazetteers and had a respectable size (i.e., it had roughly twice the number of toponyms than in previously developed corpora).

Aside from proposing a new method for toponym resolution, Gritta et al. [10] introduced GeoVirus, a dataset for the evaluation of geoparsing of news events covering global disease outbreaks and epidemics [10]. Place names are manually tagged and assigned Wikipedia page URLs along with their global coordinates.

More recently, yet another corpus was introduced in the context of the SemEval-2019 Task-12 Challenge, where toponym resolution systems were evaluated on scientific articles. Weissenbacher et al. [48] summarized the corpus used within this challenge. The corpus is composed of documents linked with different viruses, namely the Influenza A, B, and C virus, and the West Nile river virus. Additionally, some other documents associated with biomedical research articles were added to the corpus. The result was a dataset with fine-level toponyms that can be used to resolve name places in other scientific domains, particularly related to the domain of epidemiology.

Table 5.1 shows a statistical characterization of the main corpora used in the development of this project. Additionally, the number of distinct HEALPix classes considered for each dataset is also presented. Figure 5.1 shows the geographical distribution of toponyms in each of the previously explained corpora. Both the GeoVirus and the dataset developed in the context of the SemEval-2019 Task-12 Challenge have significantly more widely spread examples than the other datasets explored, that are essentially centered in the United States.

5.2 Experimental Methodology

I decided to simulate the conditions of the experiments developed by previous methods, thus making the comparisons as trustworthy as possible. In particular, I have kept the changes performed by Kulkarni et al. [18] in the LGL and the GeoVirus corpora. As previously explained in Section 3, the authors decided to unify the true coordinates corresponding to the same target entity. Regarding the WOTR corpus, I chose to use the same data split presented by the authors [5]. As far as the SpatialML is concerned, I decided to split the data in the following proportions: 90% of the instances for train and 10% for test. Finally, and regarding the corpus used in the SemEval-2019 Task-12 Challenge, I decided to merge the train and validation splits into one [48]. Since the test data was not made publicly available, I chose to split the merged data in the following proportions: 90% of the instances for train and 10% for test. The number of train and test instances for each dataset is also described in Table 5.1.

To assess the performance of the developed system across the multiple datasets, the distance between the predicted coordinates and the ground-truth coordinates is computed, using Vincenty's formula (i.e., a well-known method for calculating geodesic distances between a pair of latitude/longitude points on an ellipsoidal model of the Earth). Having these values, some evaluation metrics can be computed, such as the mean and median error distances, and an accuracy (i.e., percentage of correct decisions) based on a given threshold over the distance. Previous studies have used 161 kilometers as the threshold distance value for the accuracy, so that value is also taken into consideration in this project.

5.3 The Obtained Results

The architecture presented in Section 3.1 is the one used in my experiments. I decided to use two different pre-trained language models, namely the BERT [7] and the RoBERTa [26] models. As briefly explained in Section 3.1, BERT is probably the most widely known Transformer-based language model. However, several new language models based on the BERT architecture, namely the RoBERTa model, have been producing prosperous results in several NLP tasks. Additionally, and besides testing two different models, I also test different versions of those models (i.e., a single pre-trained language model can have multiple versions that share the same general architecture but differ in the number of parameters and in the corpora used to train the model). Particularly, the versions used in the experiments are the BERT-base and the RoBERTa-large. In brief, the BERT-base model has 12 encoder layers stacked on top of each other whereas RoBERTa-large has 24 layers of encoders stacked on top of each other. As the number of layers is increased so does the number of parameters and the number of attention heads. BERT-base has a total of 12 attention heads and 110 million parameters. On the other hand, RoBERTa-large has 16 attention heads with 355 million parameters. BERT-base has 768 hidden layers while RoBERTa-large has 1024 hidden layers.

Dataset	Mean dist. (km) ↓	Median dist. (km) ↓	Accuracy@161 km (%) ↑
SpatialML Corpus			
Learning to Rank [39]	**140**	28.71	-
My model with BERT-base	206	**9.08**	**90.9**
LGL Corpus			
MLG w/o Gaz [18]	1407	-	53.0
MLG with Gaz [18]	620	-	73.0
My model with BERT-base	**193**	**12.52**	**85.0**
WOTR Corpus			
TopoCluster [5]	604	-	57.0
TopoClusterGaz [5]	468	-	72.0
Cardoso et al. [3] model	164	11.48	81.5
My model with BERT-base	**99**	**11.11**	**87.1**
SemEval-2019 Task-12 Challenge corpus			
My model with BERT-base	**146**	**11.71**	**85.4**
GeoVirus			
MLG w/o Gaz [18]	1690	-	49
MLG with Gaz [18]	**276**	-	**85**
My model with BERT-base	720	180.80	49.8

Table 5.2: Experimental results with the base model

I have tested both the BERT-base and the RoBERTa-large language models with and without integrating geophysical properties, originating the following models: (1) BERT-base W/GeoProperties (this particular model will be addressed as the baseline) and BERT-base w/o GeoProperties, and (2) RoBERTa-large W/GeoProperties and RoBERTa-large w/o GeoProperties.

Table 5.2 summarizes the results obtained by my base model (i.e., the BERT-base model considering GeoProperties), comparing them against the results reported on previous publications that have used the same datasets and the same evaluation metrics. One can verify that my base model achieves results that outperform those of previous methods achieving state-of-the-art results in several metrics. In particular, the model obtains best results on both the WOTR and the LGL datasets, outperforming the previous best results. Regarding the SpatialML corpus, the learning to rank system from Santos et al. [39] achieves the smallest mean distance error, even though my model produces a much smaller median distance error. As to the corpus used in the SemEval-2019 Task-12 Challenge I couldn't compare the results achieved by my model with the ones obtained by the teams competing in the contest, given the fact that I didn't have access to the test data. Finally, and regarding the GeoVirus dataset, the proposed model achieves worst results than the system that relies on gazetteer information proposed by Kulkarni et al. [18]. However, when compared to the system that does not rely on gazetteer information proposed by the same authors, my model produces a much smaller mean distance error and slightly better accuracy. One possible explanation for this results is the fact that the GeoVirus dataset is mainly composed by large populated toponyms (i.e., most of the documents are concerned with locations with large populations), thus a system that relies on gazetteer information can somewhat benefit from this, by leaning to predict more common toponyms (i.e., when dealing with toponym disambiguation, a gazetteer-based system will likely opt to choose the larger toponym).

Model and Dataset	Mean dist. (km) ↓	Median dist. (km) ↓	Accuracy@161 km (%) ↑
SpatialML Corpus			
BERT-base W/GeoProperties	206	**9.08**	90.9
BERT-base w/o GeoProperties	205	**9.08**	90.6
RoBERTa-large W/GeoProperties	**182**	**9.08**	**92.1**
RoBERTa-large w/o GeoProperties	**182**	**9.08**	91.9
LGL Corpus			
BERT-base W/GeoProperties	193	12.52	85.0
BERT-base w/o GeoProperties	192	12.57	84.5
RoBERTa-large W/GeoProperties	**171**	12.26	87.8
RoBERTa-large w/o GeoProperties	172	**12.24**	**88.9**
WOTR Corpus			
BERT-base W/GeoProperties	99	11.11	87.1
BERT-base w/o GeoProperties	105	11.26	85.7
RoBERTa-large W/GeoProperties	**74**	**10.99**	**88.5**
RoBERTa-large w/o GeoProperties	81	**10.99**	88.0
SemEval-2019 Task-12 Challenge corpus			
BERT-base W/GeoProperties	146	11.71	85.4
BERT-base w/o GeoProperties	152	11.90	84.3
RoBERTa-large W/GeoProperties	**110**	**11.12**	**89.1**
RoBERTa-large w/o GeoProperties	116	11.15	87.8
GeoVirus			
BERT-base W/GeoProperties	720	180.80	49.8
BERT-base w/o GeoProperties	652	138.65	51.6
RoBERTa-large W/GeoProperties	621	32.21	59.9
RoBERTa-large w/o GeoProperties	**583**	**29.52**	**61.1**

Table 5.3: Experimental results with different modelling approaches

Additionally, it is also worth mentioning that the loss function computed over the class probability differs from corpus to corpus. In the case of the WOTR and the SemEval-19 Task-12 Corpus I use the asymmetric loss function, and in the rest of the corpora I decided to stick with the cross entropy loss function. This decision is based on the fact that both the WOTR and the corpus from the SemEval-19 Task-12 Corpus have the highest number of distinct HEALPix classes (see Table 5.1), thus could benefit from the use of the asymmetric loss function.

Table 5.3 describes the results obtained by the different modelling approaches. The results show that the large version of the RoBERTa language model obtains better results across all datasets (i.e., on average, less 42 kilometers of mean distance error, less 50 kilometers in the median error, and an increase of 3.74% on the accuracy@161) . Although the large version can lead to significantly better results, it also exponentially increases the training time, and the amount of computational resources needed to fine-tune the model. Thus, using or not a larger version of a pre-trained language model is a trade-off between better performance overall and less computational efficiency.

With respect to the experiments with geophysical information, I recorded some inconsistencies in the results. My model achieves worst results in the GeoVirus dataset, and slightly better results among the remaining corpora. In particular, I noticed a bigger gap in the results on the WOTR dataset. This can probably be explained by the fact that the WOTR dataset is mainly composed of annotated documents from a set of American Civil War archives that contain a lot of geophysical information, such as terrain

Corpus	Lowest distance error (km)	Highest distance error (km)
WOTR	Mexico (0.62) Spring River (0.70) Owen's Big Lake (1.08)	Shelter Cove (4016.65) Fort Sheward (3956.01) Gravely Ford (3179.59)
LGL	Iowans (1.16) Pa. (2.30) Pennsylvania (2.30)	Nigeria (9304.27) Philippines (8344.62) Vietnam (7674.42)
SpatialML	Tokyo (0.44) Lusaka (2.38) Basra (2.76)	Dunblane (15062.31) Cayman Islands (12900.50) British Colombia (7361.93)
SemEval-2019 Task-12 Challenge corpus	Dominican Republic (0.42) Japan (0.63) Stratford (0.983)	Topografov River (4584.17) Antananarivo (3122.57) United Arab Emirates (2399.63)
GeoVirus	United States (4.53) China (6.07) California (10.23)	New Zealand (16610.02) Wisconsin (12859.72) North America (8098.64)

Table 5.4: Toponyms with the lowest and highest distance errors

properties, water coverage, and resources, hence useful features when predicting the actual values of the geophysical properties. One can then conclude that the model indeed benefits from the addition of geophysical information, though not consistently.

Table 5.4 presents the place names with the lowest and highest distance error predictions for all corpora. There are a few conclusions one might retrieve from this statistical characterization, namely: (1) the model can identify different textual names as the same real location (e.g., *Pa.* and *Pennsylvania* both refer to the state in the United States), (2) the model can identify demonyms (i.e., a noun used to denote the natives or inhabitants of a particular country, state, city, etc.), such as *Iowans* (i.e., a native or inhabitant of the US state of Iowa), and predict the locations of the place itself, (3) there are a few small places that are among the locations with the lowest mean distance error, such as *Owen's Big Lake*, *Spring River* or *Stratford*. These results show that language model-based systems can effectively predict the geographical coordinates of some toponyms that would be poorly predicted by gazetteer-based approaches.

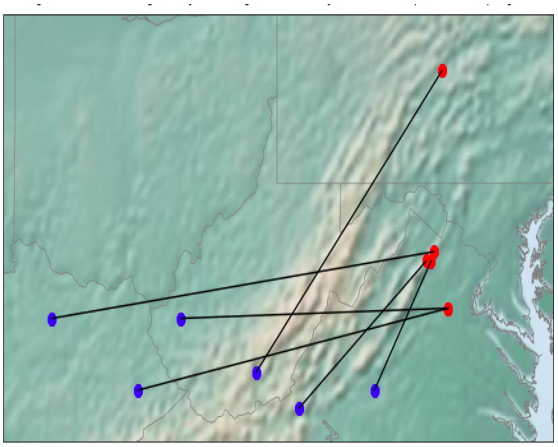
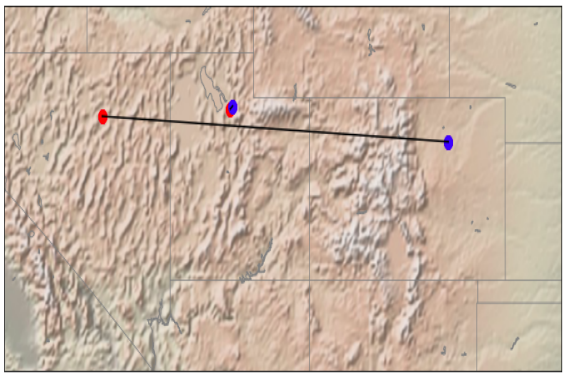
Finally, Table 5.5 presents some illustrative examples of toponym resolution together with the corresponding textual reference. Each of the examples has the document text with the place names references highlighted in red, and the image that show the real location of the place name (in red), and the corresponding predicted location (in blue). Additionally, the distance between the real and predicted locations is represented through a black line. In the examples shown, two different behaviours of the model are illustrated. In the first example, the distance between the predicted and the real locations is significant. In particular, the average distance error is 375 kilometers. Additionally, the first example has quite a few interesting toponyms such as *Peidmont* and *Peidmont Valley*, that correspond to the

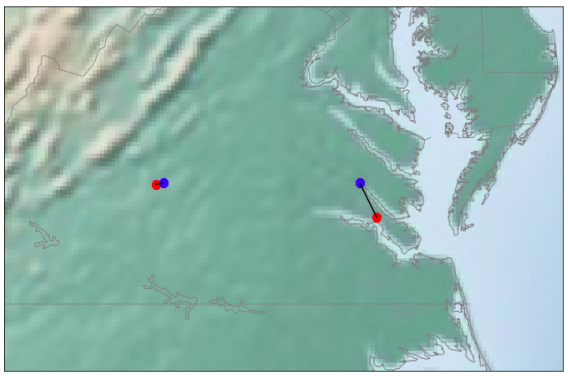
same real location and are predicted differently by the model. One possible explanation is the fact that the word *Valley* can lead the model to a different location. One more interessting location is the *Paris* toponym, a highly ambiguous place name, that is predicted incorrectly by the model. In the second example, the model predicts correctly the toponym *Salt Lake*, but misses to predict the toponym *Gravelly Ford*. This behaviour is probably explained by the fact that the context surrounding the toponym *Gravelly Ford* is not informative about its real location. In the third example, the average distance error is much smaller (i.e., 18 kilometers), probably due to the presence of toponym co-ocurrence consecutively (i.e., *Fort Magruder, Va.*), which can give clues about both toponym locations. Finally, the last example shows the same place name reference *Paris* but from a different dataset (i.e., this example is extracted from the LGL dataset, while the first three examples were extracted from the WOTR dataset), and another highly ambiguous location, *Texas*. In this example, both the *Paris* and *Texas* locations are correctly predicted, probably due to the context surrounding both toponyms (e.g., the word *Northeast* is an incredibly informative clue about the true locations of the toponyms to be disambiguated). These results show that the model is highly dependent on the context surrounding the target toponym.

5.4 Overview

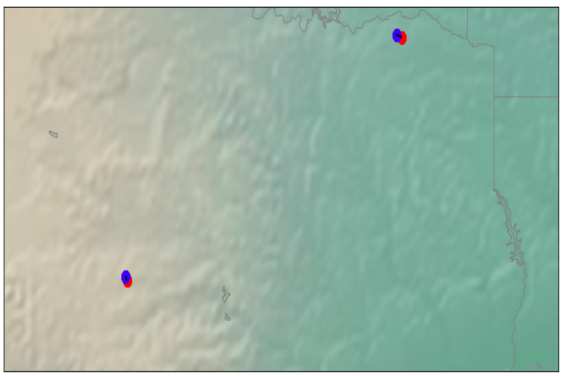
This chapter detailed the experimental evaluation. Section 5.1 characterized the corpora used throughout the experiments, while Section 5.2 presented the methodology used to perform the experiments. Finally, Section 5.3 presented the obtained results.

Table 5.5: Illustrative examples

	I took the main column on the Peidmont. At that point I sent Captain Hart with 150 men of the First New Jersey Cavalry to pass through Peidmont Valley and stop at Paris until I arrived. With 100 men of the First Pennsylvania, under Captain McGregor, and 50 men of the Third Pennsylvania, under Captain Wetherill, I marched to Markham Station in Manassas Gap. From that point I crossed the mountains by a by-path, and joined the other parties at Paris at 12 o'clock on the day of the 18th. The column under Lieutenant Bradbury lost their way and came into Paris without passing through Upperville, and captured some horses and arms without seeing any of the enemy. The column under Captain Hart passed through Piedmont Valley, and surprised and captured 15 of Mosby's guerrillas and furloughed soldiers, and a quantity of arms, equipments, and horses. The other column with myself passed into Manassas Gap to Markham, and furloughed soldiers, and a quantity of arms, equipments, horses, and some medical stores. The latter we destroyed. As we came near Paris about 40 guerrillas charged on my rear guard. I sent a squadron and charged, scattering them. No casualties on our side.
	I think a portion of the command destined for Salt Lake should halt in the neighborhood of Gravelly Ford until the emigration has passed. EXECUTIVE DEPARTMENT, Carson City, September 15, 1862. Brigadier-General WRIGHT, U. S. Army, Commanding Pacific Department: SIR: I have seen an order issued from the Department at Washington in relation to the treatment of prisoners who speak disrespectfully of the Government. I see likewise that you are about to appoint a military commission to act upon and decide the several cases arising in this department. How are we to bring such cases before said commission? Is there power; if so, where does it exist, to transport them across the mountains? Can we have a commission appointed for this Territory? Treason is very openly spoken here now since Colonel Connor's proclamation. The trouble lies here in the fact that there is only one company stationend at the fort, and they can raise a force any day more than sufficient to overpower them.



GENERAL: The following dispatch, in cipher, just received from General Kilpatrick, dated Fort Magruder, Va., March 3, 1864: HEADQUARTERS CAVALRY EXPEDITION, March 3, 1864-9 p. m. Major General A. PLEASONTON, Commanding Cavalry Corps: I have reached General Butler's lines with my command in good order. HEADQUARTERS CAVALRY CORPS, March 4, 1864. GENERAL: The following dispatch, in cipher, just received from General Kilpatrick, dated Fort Magruder, Va., March 3, 1864: HEADQUARTERS CAVALRY EXPEDITION, March 3, 1864-9 p. m. Major General A. PLEASONTON, Commanding Cavalry Corps: I have reached General Butler's lines with my command in good order. I have failed to accomplish the great object of the expedition, but have destroyed the enemy's communications at various points on the Virginia Central Railroad; also the canal and mills along the James River, and much other valuable property. Drove the enemy into and through his fortifications to the suburbs of Richmond.



Paris PD: Gang initiation text is a rumor. Paris PD: Gang initiation text is a rumor. A rumor spread across Northeast Texas through text messaging warning people to stay away from Walmart because a gang initiation is supposedly going to take place involving the killing of three women in a Walmart parking lot has no credibility, according to Paris Police Department officials. As far as we can tell, it sounds like a hoax, said Matt Birch, PPD crime prevention officer. Theres no reason to stay away from Walmart. The PPD said in a release that some people in the community, mostly young people from what the police department has been able to establish, received the message and passed it on to others in the community. The text message does not specify a particular Walmart, so each community thinks it is the store in their community, the release stated. We do not believe the text message is factual, Birch said. They are getting hit all over the Metroplex with it. Its a Northeast Texas thing, not just a Paris thing.

6

Conclusions and Future Work

Contents

In this dissertation, I proposed a novel method using a Transformer-based pre-trained language model for the toponym resolution task, by considering multiple textual inputs, producing multiple outputs for classification and regression tasks. More specifically, the neural network predicts a probability distribution over HEALPix regions, and then uses this probability distribution to calculate the corresponding geographical coordinates of the toponym to be disambiguated. Additionally, I conducted several experiments, including the use of large versions of pre-trained language models, and adding geophysical properties retrieved from a raster dataset to guide the prediction of geographical coordinates.

The proposed method was tested on the WOTR, the LGL, and the SpatialML. Additionally, I also tested the model on scientific corpora such as the corpora used in the SemEval-2019 Task-12 Challenge, and the GeoVirus corpus. The results obtained confirm the superiority of the proposed model over previous methods that produced state-of-the-art results. The use of the large version of the RoBERTa pre-trained model produces better results than the non-large version of the BERT model, even though it introduces a trade-off between performance and efficiency, given that the larger version requires major computational resources to fine-tune. The incorporation of external geographical information into the model had a generally beneficial impact on the results, even though it also produced worst results in some experiments.

6.1 Overview on the Contributions

The contributions of my M.Sc. thesis are the following:

- Proposal of an end-to-end toponym resolution model architecture relying on the fine-tuning of a pre-trained language model, building a representation from the toponym that is to be disambiguated, as well as its surrounding context.

- Use of different pre-trained language models, namely the BERT and RoBERTa models. Additionally, different versions of the same model were used. The results reported show that the large versions of the model (i.e., versions with more trainable parameters and trained on a larger set of data) obtain better, although increasing the training time and the computational resources needed to train the model.

- Additional experiments using external information from geophysical properties (i.e., land coverage, terrain elevation, and percentage of vegetation) extracted from external raster datasets and incorporated in the proposed model to guide the prediction of the geographic coordinates.

6.2 Future Work

For future work, I would like to add a term to the cost function based on a contrastive function. The idea behind it is to preserve neighborhood relationships between data instances. More specifically, since the training is done in mini-batches, the distance between the predicted coordinates among each instance in the batch is compared to the distance between the ground-truth coordinates among each instance in the batch.

Additionally, it would be interesting to explore other techniques to partition the Earth into a set of discrete cells. Some approaches that were first introduced in the context of image geolocation that could possibly be analyzed are:

- **Combinatorial Partitioning**: Hongsuck Seo et al. [13] proposed an algorithm to tackle the trade-off between accuracy and overfitting by generating a large number of fine-grained classes by intersecting multiple coarse-grained cells, allowing a model to predict locations at a fine-scale while keeping several training examples per class.
- **Hierarchical Partitioning**: An algorithm that exploits hierarchical knowledge of multiple partitions was proposed by Muller-Budack et al. [34]. This approach also uses the S2 library to generate sets of geoclasses, and applies an adaptive hierarchical subdivision, where each cell is the node of a quad-tree (i.e., a tree data structure in which each internal node has exactly four children).
- **Using an MvMF Output Layer**: Izbicki et al. [14] introduced a geolocation method that exploits the Earth's spherical geometry based on the *von Mises-Fisher* (vMF) distribution, which is one of the standard distributions in the field of directional statistics and that can be seen as the spherical analog of the Gaussian distribution.

Bibliography

[1] Amitay, E., Har'El, N., Sivan, R., and Soffer, A. (2004). Web-a-Where: Geotagging Web Content. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[2] Ben-Baruch, E., Ridnik, T., Zamir, N., Noy, A., Friedman, I., Protter, M., and Zelnik-Manor, L. (2020). Asymmetric Loss for Multi-label Classification. *arXiv preprint arXiv:2009.14119*.

[3] Cardoso, A. B., Martins, B., and Estima, J. (2019). Using Recurrent Neural Networks for Toponym Resolution in Text. In *Proceedings of the EPIA Conference on Artificial Intelligence*.

[4] DeLozier, G., Baldridge, J., and London, L. (2015). Gazetteer-Independent Toponym Resolution using Geographic Word Profiles. In *Proceedings of the AAAI conference on Artificial Intelligence*.

[5] DeLozier, G., Wing, B., Baldridge, J., and Nesbit, S. (2016). Creating a Novel Geolocation Corpus from Historical Texts. In *Proceedings of the Linguistic Annotation Workshop held in conjunction with Association for Computational Lingustics*.

[6] Delpeuch, A. (2019). OpenTapioca: Lightweight Entity Linking for WikiData. *arXiv preprint arXiv:1904.09131*.

[7] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

[8] Goldberg, Y. (2017). Neural Network Methods for Natural Language Processing. *Synthesis Lectures on Human Language Technologies, 10(1)*.

[9] Gorski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., and Bartelmann, M. (2005). HEALPix: A Framework for High-resolution Discretization and Fast Analysis of Data Distributed on the Sphere. *The Astrophysical Journal, 622(2)*.

[10] Gritta, M., Pilehvar, M., and Collier, N. (2018). Which Melbourne? Augmenting Geocoding with Maps. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[11] Grover, C., Tobin, R., Byrne, K., Woollard, M., Reid, J., Dunn, S., and Ball, J. (2010). Use of the Edinburgh geoparser for georeferencing digitized historical collections. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 368(1925)*.

[12] Halterman, A. (2017). Mordecai: Full Text Geoparsing and Event Geocoding. *Journal of Open Source Software*.

[13] Hongsuck Seo, P., Weyand, T., Sim, J., and Han, B. (2018). CPlaNet: Enhancing Image Geolocalization by Combinatorial Partitioning of Maps. In *Proceedings of the European Conference on Computer Vision*.

[14] Izbicki, M., Papalexakis, E. E., and Tsotras, V. J. (2019). Exploiting the Earth's Spherical Geometry to Geolocate Images. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*.

[15] Karimzadeh, M., Huang, W., Banerjee, S., Wallgrün, J. O., Hardisty, F., Pezanowski, S., Mitra, P., and MacEachren, A. M. (2013). GeoTxT: a Web API to Leverage Place References in Text. In *Proceedings of the Workshop on Geographic Information Retrieval*.

[16] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*.

[17] Kolitsas, N., Ganea, O.-E., and Hofmann, T. (2018). End-to-End Neural Entity Linking. *arXiv preprint arXiv:1808.07699*.

[18] Kulkarni, S., Jain, S., Hosseini, M. J., Baldridge, J., Ie, E., and Zhang, L. (2021). Multi-Level Gazetteer-Free Geocoding. In *Proceedings of Second International Combined Workshop on Spatial Language Understanding and Grounded Communication for Robotics*.

[19] Leidner, J. L. (2007). Toponym Resolution in Text: Annotation, Evaluation and Applications of Spatial Grounding. In *ACM SIGIR Forum*.

[20] Leidner, J. L., Sinclair, G., and Webber, B. (2003). Grounding spatial named entities for information extraction and question answering. In *Proceedings of the HLT-NAACL Workshop on Analysis of Geographic References*.

[21] Li, H., Wang, M., Baldwin, T., Tomko, M., and Vasardani, M. (2019). UniMelb at SemEval-2019 Task 12: Multi-model Combination for Toponym Resolution. In *Proceedings of the International Workshop on Semantic Evaluation*.

[22] Lieberman, M. D. and Samet, H. (2012). Adaptive Context Features for Toponym Resolution in Streaming News. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[23] Lieberman, M. D., Samet, H., and Sankaranarayanan, J. (2010a). Geotagging with Local Lexicons to Build Indexes for Textually-Specified Spatial Data. In *Proceedings of the IEEE International Conference on Data Engineering*.

[24] Lieberman, M. D., Samet, H., and Sankaranayananan, J. (2010b). Geotagging: Using proximity, sibling, and prominence clues to understand comma groups. In *Proceedings of the ACM Workshop on Geographic Information Retrieval*.

[25] Liu, Q., Kusner, M. J., and Blunsom, P. (2020). A Survey on Contextual Embeddings. *arXiv preprint arXiv:2003.07278*.

[26] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

[27] Loshchilov, I. and Hutter, F. (2017). Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*.

[28] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. *arXiv preprint arXiv:1508.04025*.

[29] Mani, I., Doran, C., Harris, D., Hitzeman, J., Quimby, R., Richer, J., Wellner, B., Mardis, S., and Clancy, S. (2010). SpatialML: annotation scheme, resources, and evaluation. *Language Resources and Evaluation, 44(3)*.

[30] Martins, P. H., Marinho, Z., and Martins, A. F. (2019). Joint Learning of Named Entity Recognition and Entity Linking. *arXiv preprint arXiv:1907.08243*.

[31] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the International Conference in Neural Information Processing Systems*.

[32] Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., and Gao, J. (2020). Deep learning based text classification: A comprehensive review. *arXiv preprint arXiv:2004.03705*.

[33] Mulang, I. O., Singh, K., Vyas, A., Shekarpour, S., Vidal, M.-E., and Auer, S. (2020). Encoding Knowledge Graph Entity Aliases in Attentive Neural Network for Wikidata Entity Linking. *Proceedings of the International Conference on Web Information Systems Engineering*.

[34] Muller-Budack, E., Pustu-Iren, K., and Ewerth, R. (2018). Geolocation Estimation of Photos using a Hierarchical Model and Scene Classification. In *Proceedings of the European Conference on Computer Vision*.

[35] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in neural information processing systems*.

[36] Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

[37] Radford, B. J. (2021). Regressing Location on Text for Probabilistic Geocoding. *arXiv preprint arXiv:2107.00080*.

[38] Roller, S., Speriosu, M., Rallapalli, S., Wing, B., and Baldridge, J. (2012). Supervised Text-based Geolocation using Language Models on an Adaptive Grid. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

[39] Santos, J., Anastácio, I., and Martins, B. (2015). Using machine learning methods for disambiguating place references in textual documents. *GeoJournal, 80(3)*.

[40] Sevgili, O., Shelmanov, A., Arkhipov, M., Panchenko, A., and Biemann, C. (2020). Neural Entity Linking: A Survey of Models based on Deep Learning. *arXiv preprint arXiv:2006.00575*.

[41] Shahbazi, H., Fern, X. Z., Ghaeini, R., Obeidat, R., and Tadepalli, P. (2019). Entity-aware ELMo: Learning Contextual Entity Representation for Entity Disambiguation. *arXiv preprint arXiv:1908.05762*.

[42] Smith, D. A. and Crane, G. (2001). Disambiguating Geographic Names in a Historical Digital Library. In *Proceedings of the European Conference on Research and Advanced Technology of Digital Libraries*.

[43] Speriosu, M. and Baldridge, J. (2013). Text-Driven Toponym Resolution using Indirect Supervision. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[44] Speriosu, M. A. (2013). *Methods and Applications of Text-Driven Toponym Resolution with Indirect Supervision*. PhD thesis, The University of Texas at Austin.

[45] Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved Semantic Representations from Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the Annual Meeting of the Associatioon for Computational Linguistics*.

[46] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is All You Need. In *Proceedings of the International Conference in Neural Information Processing Systems*.

[47] Wang, X., Ma, C., Zheng, H., Liu, C., Xie, P., Li, L., and Si, L. (2019). DM_NLP at SemEval-2019 Task 12: A Pipeline System for Toponym Resolution. In *Proceedings of the International Workshop on Semantic Evaluation*.

[48] Weissenbacher, D., Magge, A., O'Connor, K., Scotch, M., and Gonzalez-Hernandez, G. (2019). SemEval-2019 Task 12: Toponym Resolution in Scientific Papers. In *Proceedings of the International Workshop on Semantic Evaluation*.

[49] Weissenbacher, D., Tahsin, T., Beard, R., Figaro, M., Rivera, R., Scotch, M., and Gonzalez, G. (2015). Knowledge-driven geospatial location resolution for phylogeographic models of virus migration. *Bioinformatics, 31(12)*.

[50] Wing, B. P. et al. (2015). *Text-Based Document Geolocation and its Application to the Digital Humanities*. PhD thesis, The University of Texas at Austin.

[51] Wu, L., Petroni, F., Josifoski, M., Riedel, S., and Zettlemoyer, L. (2019). Zero-shot Entity Linking with Dense Entity Retrieval. *arXiv preprint arXiv:1911.03814*.

[52] Yamada, I., Washio, K., Shindo, H., and Matsumoto, Y. (2020). Global Entity Disambiguation with Pretrained Contextualized Embeddings of Words and Entities. *arXiv preprint arXiv:1909.00426*.

[53] Zhang, W. and Gelernter, J. (2014). Geocoding Location Expressions in Twitter Messages: A Preference Learning Method. *Journal of Spatial Information Science, 2014(9)*.

[54] Zhou, X., Liu, X., Jiang, J., Gao, X., and Ji, X. (2021). Asymmetric Loss Functions for Learning with Noisy Labels. *arXiv preprint arXiv:2106.03110*.